UNIVERSIDAD CARLOS III DE MADRID

Ph.D Thesis

Efficient Voice and Video Traffic Delivery in IEEE 802.11 WLANs: Design, Implementation and Experimental Evaluation

Author: José Pablo Salvador García
Advisor: Dr. Pablo Serrano Yáñez-Mingot, Universidad Carlos III de Madrid

DEPARTMENT OF TELEMATIC ENGINEERING

Leganés (Madrid), April 2016

E cient Voice and Video Tra c Delivery in 802.11 WLANs: Design, Implementation and Experimental Evaluation

A dissertation submitted in partial ful llment of the requirements for the degree of Doctor of Philosophy

Prepared by
Jose Pablo Salvador Garc a, Universidad Carlos III de Madrid, IMDEA Networks Institute

Under the advice of
Pablo Serrano Yanez-Mingot, Universidad Carlos III de Madrid

Departamento de Ingenier a Telematica, Universidad Carlos III de Madrid

---

Date: April 2016

This work has been supported by IMDEA Networks Institute.

TESIS DOCTORAL

Efficient Voice and Video Traffic Delivery in IEEE 802.11 WLANs: Design, Implementation and Experimental Evaluation

Autor: Jose Pablo Salvador Garc a
Director: Dr. Pablo Serrano Yanez-Mingot, Universidad Carlos III de Madrid

Firma del tribunal cali cador:

Firma:

Presidente: Ilenia Tinnirello

Vocal: Boris Bellalta Jimenez

Secretario: Paolo Casari

Cali cacion:

Leganes, 8 de abril de 2016

Acknowledgements

First and foremost I would like to thank my advisor, Dr. Pablo Serrano Yanez Mingot, whose great guidance and advice has been key in producing this thesis. He is an excellent advisor and has always been supportive, patient and has provided insightful feedback along with the best possible research environment one could ask for.

I would also like to express my gratitude to Dr. Albert Banchs, Dr. Vincenzo Mancuso and Dr. Francesco Gringoli. They are three astonishing top-researchers and people, who helped me out enormously during my thesis. I appreciate their time and e ort in reviewing my work, and for their invaluable support during this process.

A special thanks also goes to Dr. Carlos J. Bernardos Cano who showed me the PhD path and the valuable research that IMDEA Networks Institute was carrying out. Thanks to everyone at IMDEA and NETCOM, and mainly Jose Felix Kukielka and Alberto Gordillo, always there to help me out and allowed me to focus on my research.

Thanks to Dr. Edward Knightly for hosting me and giving me the opportunity to collaborate in the Rice Networks Group at Rice University. It has been an incredible experience. I am very thankful to Naren, Oscar, Ryan, Adriana, Xu and Alex Chino - you rock guys. Thanks for the wonderful time we have spent together.

A special thanks goes to Marco, Fabio, Inaki, Ignacio, Thomas, Camilo, Luca and Andres for all of their help, advice and guidance. They have been such incredible friends, who were always there for me. Also to Andra, Arash and Allyson for making this journey funnier and bearable. Thanks to Christina Vlachou and Nicolo Facchi, with whom I collaborated on the last stage of my PhD.

Others who have helped me through my journey, pushed me when I needed pushing and cheered me up during my hard times include Kirk, Rusty, Alberto, Edgar and Cris.

To Isabel, who has been my rock during this entire process and supported me with tons of love, trust and caring. Thanks for being my best friend and greatest support. We will have each others back always, and you know it. This thesis would have not been possible without you.

Last but not least, I would would like to thank my parents, Pablo and Pascuala, and my sister, Ana, for their unconditional love and help. The supportive atmosphere they provided is what has allowed me to succeed.

Abstract

While the IEEE 802.11 protocol has fostered ubiquitous connectivity for wireless users, it was not originally designed to e ciently handle voice or video tra c, which nowadays accounts for most of the Internet tra c. Voice tra c handling is extremely ine cient in existing WLANs due to the large overhead of the protocol and the time spent in contention. Video multicast streaming is both ine ective and ine cient, as its transmissions lack reliability and use a low Modulation and Coding Scheme (MCS). The more robust, but slower, transmission rates result in the well-known performance anomaly problem that degrades network performance. In this thesis we analyze, design and implement solutions to improve the network performance and e ciency when voice and video tra c are present.

We rst design and experimentally validate a simple yet very e ective scheme (*VoIPiggy*) to improve the e ciency of WLANs with voice tra c. The key idea of *VoIPiggy* is to piggyback voice frames onto MAC layer acknowledgments, hereby reducing both the frame overhead and the number of times to access the medium, i.e., the time spent in contention as stations aggregate two frames into a single transmission. To quantify the gains of our proposal, we perform an analysis in terms of capacity and delay. Our experimental and analytical results show a dramatic performance improvement, doubling the number of voice conversations that can be allocated in WLANs.

Second, we explore a set of mechanisms included in the 802.11aa standard (namely, Direct Multicast Sequence, Groupcast with Retries Unsolicited Retries and Block Acknowledgment), which allow e cient and robust multicast transmission in WLANs. To that aim, we rst perform extensive simulations to understand the trade-o s resulting from using the proposed mechanisms. Our results quantify these trade-o s in terms of robustness, resource consumption and complexity, and provide a set of recommended guidelines for their use according to the network scenario.

Finally, we develop the rst open source implementation of these mechanisms over commodity hardware. We assess their performance under a variety of real-life scenarios, and outline their e ectiveness and e ciency when delivering video tra c. Our results provide key insights on which mechanism results more appropriate for a given scenario, this being speci ed in terms of number of stations, background tra c and video bandwidth.

List of Abbreviations

AC	Access Category
ACK	Acknowledgment
AIFS	Arbitrary Inter Frame Space
AMPDU	Aggregated MPDU
AP	Access Point
AVB	Audio Video Bridging
BA	Block Acknowledgment
BAREP	BA Reply
BAREQ	BA Request
BC	Backo Counter
BEB	Binary Exponential Backo
BRS	Basic Rate Set
BSS	Basic Service Set
CAC	Call Admission Control
CBR	Constant Bit Rate
CDF	Cumulative Distribution Function
CF	Contention-Free
CFP	Contention-Free Period
COTS	Commercial O -The-Shelf
CP	Contention Period
CW	Contention Window
DCF	Distributed Coordination Function
DIFS	Distributed Inter Frame Space
DMA	Direct Memory Access

DMS	Direct Multicast Sequence
EDCA	Enhanced Distributed Channel Access
ELBP	Enhanced Leader Based Protocol
FEC	Forward Error Correction
GATS	Group Addressed Transmission Service
GCR	Groupcast with Retries
HCCA	HCF Controlled Channel Access
HCF	Hybrid Coordinated Function
IEEE	Institute of Electrical and Electronics Engineers
IFS	Inter Frame Space
IP	Internet Protocol
IGMP	Internet Group Management Protocol
LAN	Local Area Network
LBP	Leader Based Protocol
MAC	Medium Access Control
MCS	Modulation and Coding Scheme
MPDU	MAC Protocol Data Unit
MSRP	Multiple Stream Reservation Protocol
NAT	Network Address Translation
OBSS	Overlapping BSS
OFDM	Orthogonal Frequency-Division Multiplexing
PCF	Point Coordinated Function
PHY	Physical Layer
PTP	Precision Time Protocol
QoE	Quality of Experience
QoS	Quality of Service
RTT	Round Trip Time

SCS	Stream Classi cation Service
SIFS	Short Inter Frame Space
SRP	Stream Reservation Protocol
TCP	Transport Control Protocol
TDMA	Time Division Multiple Access
TXOP	Transmission Opportunity
UDP	User Datagram Protocol
UR	Unsolicited Retries
VoIP	Voice over IP
VoLTE	Voice over LTE
WiFi	IEEE 802.11 WLAN
WLAN	Wireless Local Area Network
WMM	WiFi Multimedia

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Wireless communications have in the IEEE 802.11 standard for WLAN [1] its greatest representative, which has become one of the most common technologies to provide ubiquitous Internet access. The notoriety of this technology is endorsed by the continuous development of solutions that enhance the 802.11 network performance and allow to o er an optimal service to a larger number of users in a more e cient way. This thesis addresses the problematic of serving real-time multimedia tra c, which represents a huge portion of the total Internet tra c, in 802.11 WLANs.

In this chapter we expose the main motivations and reasons behind this thesis. Next, we present the main contributions of our work and the structure of the thesis. Finally, we include all the publications which support this work.

1.1 Motivation

A fundamental reason behind the success of 802.11, namely WiFi, is its inexpensive cost as it is easy to deploy and operates in unlicensed frequency bands. Proof of this success is the number of devices with WiFi capabilities foreseen to reach up to 1.9 Billions by 2019, exceeding signi cantly the number of cellular devices (542 Millions) [2]. Along with its pervasive deployment, the representative growth in mobile cellular data tra c has been mitigated by o oading more than half of the tra c from mobile devices to the xed network by means of WiFi or femtocell access [2]. This trend is foreseen to continue in spite of the evolution of mobile networks. Without WiFi o oading, the growth of the tra c demand in cellular networks would have become unsustainable as [2] indicates that mobile data tra c would have grown 84% rather than 69% in 2014. Voice and video streaming tra c represent a signi cant portion of the overall Internet tra c. In the case of voice over WiFi, its minutes of use will account for more than half of all mobile IP voice tra c [2]. Globally, IP video will represent 80% of all the tra c by 2019 according to the forecast in [3].

802.11 is being commoditized for voice communications, also known as Voice over IP (VoIP), with the proliferation of portable devices running voice applications, such as Skype or Google Hangouts. The creation of a certi cation program devoted to VoIP tra c over WiFi networks[1] con rms its importance and the need for increasing performance in voice transmission. Together with the widespread deployment of high-rate access points (APs), with modulation schemes reaching up to 7 Gb/s as de ned by the new amendments 802.11ac [4] and 802.11ad [5], are also enabling the introduction of Full HD video applications with relatively large bandwidth demands, like YouTube, Skype videoconferencing or VideoLAN video-streaming. Another proof of the proliferation of multimedia tra c is that the WiFi Alliance included in its certi cation programs a WiFi Multimedia (WMM) certi cate, back in 2004. This program introduces Quality of Service (QoS) concepts and prioritization of tra c classes in WiFi, di erentiating among voice, video, best-e ort, and background data. Lately, admission control mechanisms were introduced to enhance the bandwidth management and the prioritization de ned in WMM.[2] Consequently, IEEE 802.11 networks have to deal with an important amount of voice and video tra c, thereby **this thesis will focus on improving the performance for these two types (voice and video multicast) of tra c in 802.11 networks**. Speci cally, as the original de nition of the standard was not designed for the special characteristics of this particular tra c, we address several ine ciencies and provide solutions to improve performance, including experimental evaluations in commodity hardware.

Voice tra c is transmitted in small frames (60-160 bytes), making operation of the legacy IEEE 802.11 DCF extremely ine cient as the payload size is comparable to the packet header size. In addition, the voice quality is highly vulnerable to the presence of data cross-tra c. The overhead ine ciency is not alleviated by introducing higher data rates, since these do not change the protocol overhead and hence do not signi cantly reduce the fraction of time wasted due to the 802.11 backo mechanism. Voice quality vulnerability can be reduced by means of prioritization, which is introduced by the EDCA mechanism presented in the 802.11e standard [6]. However, this still incurs substantial overhead and does not solve the problem of vulnerability to legacy tra c [7]. As a result, the performance of data tra c has to be reduced to sustain enough quality for voice tra c.

Motivated by the low e ciency of the standard operation with voice tra c, in this dissertation we propose a simple yet e ective mechanism, namely *VoIPiggy*, to reduce the overhead of the Medium Access Control (MAC) operation, which also reduces the time that a station spends contending for the wireless medium. This mechanism piggybacks voice frames onto MAC Acknowledgment (ACK)s reducing both the MAC overhead and

[1] `https://www.wi-fi.org/file/wi-fi-certified-voice-enterprise-delivering-wi-fi-voice-to-the-enterprise-2012`

[2] WiFi Multimedia (WMM) certi cate: `https://www.wi-fi.org/discover-wi-fi/wi-fi-certified-wmm-programs`

the average number of stations contending for channel access, which improves overall system performance. As a result, VoIPiggy increases the number of voice and data ows that can be served, and improves the delay performance of voice tra c.

For the case of video multicast streaming, the original 802.11 standard was poorly suited for e cient support of multimedia ows, and in particular video streams, for several reasons: (*i*) the originally supported physical transmission rates (1 and 2 Mb/s) imposed a severe bottleneck on the maximum achievable rate, regardless of the e ciency of the MAC protocol; (*ii*) only *best-e ort* service was supported, thus preventing any sort of tra c di erentiation that could improve the performance of multimedia applications, which as compared to data tra c can relax their reliability requirements below 100 % to improve performance; (*iii*) multicast transmissions were very ine cient and unreliable (as widely reported in various works, see e.g. [8]), which eventually prevented its use on WLANs, as there is no proper support for simultaneous video streaming to various receivers. The subsequent amendments to the 802.11 standard have lessened the rst two limitations. Newer amendments provide a large increase in terms of achievable bandwidth, however, the new demands due to the arrival of high de nition video impose a burden on the current MAC scheme, in particular for the delivery of multicast tra c. The 802.11e amendment [6] introduced tra c di erentiation via the setting of the contention parameters, thus enabling both the ability to prioritize one type of tra c over other and a more e cient operation of WLANs by properly tuning the MAC parameters [9]. Although support for unicast tra c has vastly been improved by means of the aforementioned MAC/PHY amendments, support of multicast tra c over 802.11 WLANs still remains highly ine cient. Therefore, the remaining challenge is to e ciently support multicast over 802.11 WLANs.

The IEEE 802.11aa [10] has recently addressed this last limitation, with the de nition of various mechanisms to support *Robust streaming of Audio Video Transport Streams*. Its focus is to extend the 802.11 standard [1] with mechanisms tailored to improve performance of multimedia streaming over WLAN. In particular, the new mechanisms target at signi cantly improving both the e ectiveness in terms of reliability and the e ciency of the delivery of multicast tra c.

In this thesis, we present a thorough simulation-based evaluation of the groupcast mechanisms de ned in the 802.11aa standard, named GATS, in order to assess their performance in a scenario with real video streams. We quantify both the reliability that each of the considered mechanisms provides to a video multicast stream, for di erent numbers of receivers and radio channel conditions, as well as the e ciency in terms of the overhead introduced by each mechanism. In addition, we develop the rst experimental assessment of GATS, by implementing the new mechanisms over commodity 802.11 cards and giving guidelines on the optimal con guration of the GATS parameters for a wide set of scenarios.

1.2 Summary of thesis contributions

The contributions of this thesis are summarized as follows. First, we propose a novel mechanism, VoIPiggy [11, 12], which dramatically improves the overall performance of WLANs when voice tra c is present. It basically consists on piggybacking voice frames onto layer 2 ACKs in the uplink direction. By embedding voice frames into MAC ACKs, this approach reduces the MAC overhead given the small size of voice frames and the number of stations contending for channel access. As a result, VoIPiggy boosts the number of concurrent voice calls and data stations present in the network. Our scheme is compatible with the IEEE 802.11 standard as it requires only minor changes at the AP and it can serve legacy and VoIPiggy stations simultaneously. We conduct a theoretical analysis of the throughput, the capacity (i.e., the number of voice and data ows that can be supported) and the delay performance of VoIPiggy operating in a WLAN. We implement VoIPiggy in Commercial O -The-Shelf (COTS) devices and validate its operation against the legacy 802.11 operation. The implementation of our scheme introduces modi cations at the driver and rmware levels. We provide an extensive performance evaluation in a testbed of 30 nodes. These experiments con rm the accuracy of the analytical model and that VoIPiggy achieves dramatic performance improvements, practically doubling the capacity of the WLAN.

Second, we present a thorough simulation-based evaluation of the groupcast mechanisms de ned in the current version of the 802.11aa standard [13], in order to assess their performance in a scenario with real video multicast streams. We quantify both the reliability that each of the considered mechanisms provides to a video multicast stream, for di erent numbers of receivers and radio channel conditions, as well as the e ciency in terms of the overhead introduced by each mechanism with respect to the scheme that consumes the least amount of resources.

Finally, we implement the GATS mechanisms over COTS hardware, reporting the complexity and new functionality required to implement each mechanism. To enable researchers experimentation with the new schemes, the source code of our implementation is publicly available at `http://www.ing.unibs.it/openfwwf/GATS.php`. We deploy a test-bed with 30 GATS-enabled nodes, validating the implementation through extensive measurements that con rm the e ciency of the prototype. We conduct extensive experiments under a variety of conditions with real video tra c in scenarios with di erent numbers of receivers and data stations, assessing the quality of the video received and the resources left for data tra c, and con rming the impact of the con guration parameters of some of the schemes.

1.3 Thesis overview

The rest of this thesis is structured as follows. Chapter 2 summarizes the operation of the IEEE 802.11 protocol and presents the relevant literature related to this thesis. In Chapter 3 we first introduce the VoIPiggy mechanism, and then we analyze its performance. We also describe the implementation of the proposed scheme and its functional modules. Then, we derive an analytical model for the delay and throughput and experimentally validate its performance in a wide set of network conditions. In Chapter 4 we first present an overview of the mechanisms (legacy and novel) available for the transmission of video flows using IEEE 802.11. Next we detail the operation of a set of the mechanisms introduced by 802.11aa, namely GATS, which constitutes one of the cornerstones of this thesis. Then, we present a performance assessment and comparison of the different schemes in terms of reliability, overhead and cost per service under a variety of scenarios, and for a given quality threshold. In Chapter 5 we describe the implementation of GATS over the 802.11 open-source platform that we use and report the experimental assessment of GATS. Finally, Chapter 6 concludes this dissertation with the final remarks and futures lines of work.

1.4 Publications

The research performed in this dissertation resulted in four conference papers, three journal articles and one magazine article. We collaborated with many researchers within the course of this dissertation, especially in the framework of EU FP7 FLAVIA[3] project. This thesis covers contributions from the following literature [11 17]:

P. Serrano, **P. Salvador**, V. Mancuso, Y. Grunenberger, Experimenting with Commodity 802.11 Hardware: Overview and Future Directions, in IEEE Communications Surveys & Tutorials, vol.17, no.2, pp.671-699, Second quarter 2015 (Chapter 2).

P. Salvador, S. Paris, C. Pisa, P. Patras, Y. Grunenberger, X. Perez-Costa, J. Gozdecki, A Modular, Flexible and Virtualizable Framework for IEEE 802.11, in *Proc. of Future Network & Mobile Summit (FutureNetw)*, Berlin, Germany, July 2012 (Chapter 2).

P. Salvador, F. Gringoli, V. Mancuso, P. Serrano, A. Mannocci, A. Banchs, VoIPiggy: Implementation and evaluation of a mechanism to boost voice capacity in 802.11 WLANs, in *Proc. of IEEE International Conference on Computer Communications (INFOCOM) mini-conference*, Orlando, FL, March 2012 (Chapter 3).

[3] `http://www.ict-flavia.eu/`

P. Salvador, V. Mancuso, P. Serrano, F. Gringoli, A. Banchs, VoIPiggy: Analysis and Implementation of a Mechanism to Boost Capacity in IEEE 802.11 WLANs Carrying VoIP Tra c, in *IEEE Transactions on Mobile Computing*, vol.13, no.7, pp.1640-1652, July 2014 (Chapter 3).

A. De La Oliva, P. Serrano, **P. Salvador**, A. Banchs, Performance Evaluation of the IEEE 802.11aa Multicast Mechanisms for Video Streaming, in *Proc. of IEEE International Symposium and Workshops on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, Madrid, Spain, June 2013 (Chapter 4).

P. Salvador, L. Cominardi, P. Serrano, F. Gringoli, A rst implementation and evaluation of the IEEE 802.11aa group addressed transmission service, in *ACM SIGCOMM Computer Communication Review*, vol. 44(1), pp. 35-41, July 2014 (Chapter 4, Chapter 5).

P. Salvador, F. Gringoli, P. Serrano, N. Facchi, S. Paris, Making a Case for Flexible 802.11 Architectures, in *Proc. of IEEE International Conference on Communications (ICC)*, London, United Kingdom, June 2015 (Chapter 2, Chapter 5).

Additionally to the above, the following paper [18] with related content has been published during the development of this thesis:

C. Vlachou, A. Banchs, **P. Salvador**, J. Herzen, P. Thiran, Analysis and Enhancement of CSMA/CA with Deferral in Power-Line Communications, in *IEEE Journal on Selected Areas in Communications (JSAC)*, 2016 *Accepted for publication.*

Chapter 2

Background and Related Work

In this chapter, rst we detail the basic operation of the 802.11 protocol and give an overview of the existing 802.11 amendments. Next, we introduce the video multicast streaming operation within 802.11. Finally, we present the related work of the voice and video tra c in 802.11 networks.

2.1 Basic Operation of 802.11

Here we introduce the protocol followed by 802.11 MAC and focus on the issues directly related to voice and video transmission. The operation of 802.11 has not radically changed over the years, mainly to ensure backwards compatibility.[1] Some mechanisms have been proposed in 802.11e and 802.11aa to handle speci cally voice and multicast video streaming tra c, but there are still problems to be tackled, which we address in this thesis. In the case of voice, 802.11e prioritizes the voice and video tra c, but still leaves some room for improvement in terms of multicast reliability and control overhead for the voice tra c. In the case of multicast video streaming, 802.11aa proposes some new mechanisms to handle it, although there does not exist any implementation.

2.1.1 Overview of the 802.11 MACs

The 802.11 standard de nes two di erent channel access mechanisms, a centralized one, namely Point Coordinated Function (PCF), and a distributed one, knows as DCF. Most of the currents WLANs devices only support DCF, most widely used (and now, legacy), which involves a CSMA/CA access scheme that retransmits MAC Protocol Data Units (MPDUs) through a stop-and-wait mechanism. The basic operation of DCF is illustrated in Figure 2.1. More speci cally, when an MPDU is ready for transmission, the hardware sets a Backo Counter (BC) to a uniformly distributed value, limited by the Contention Window (CW), to CW 1. Then, it waits for the channel to be idle

[1] We refer the interested reader to [19] for an excellent overview of IEEE 802.11 amendments.

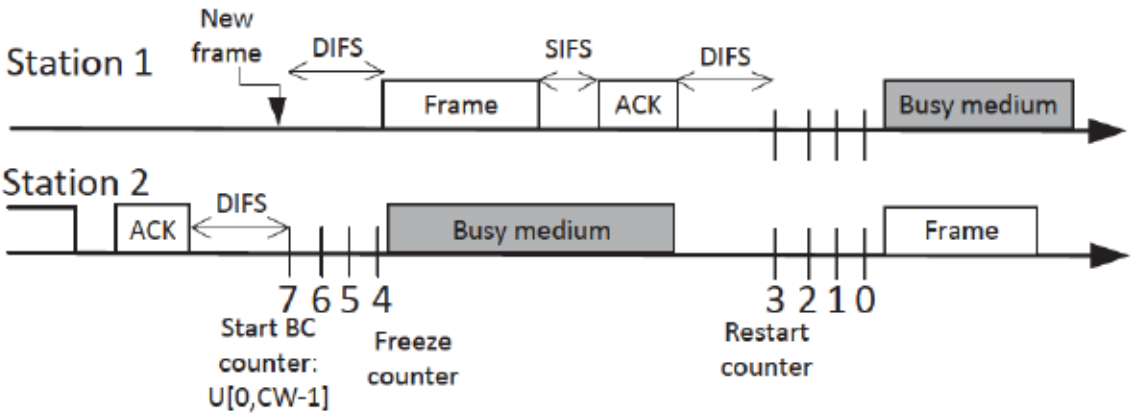


Figure 2.1: Basic operation of the DCF mechanism.

for a Distributed Inter Frame Space (DIFS) time, to start decrementing the BC at every *time slot*, eventually transmitting the MPDU when BC reaches zero. If the destination successfully acknowledges the MPDU by sending an ACK frame after a Short Inter Frame Space (SIFS), then the CW is reset to the minimum CWmin, otherwise it is doubled (until a maximum CWmax is reached). All the traffic is unified into one single class, hereby DCF only provides a best effort service.

802.11e amendment introduces a new mechanism, namely EDCA, which provides concurrent channel access via four independent queues of different access priorities, each one with a specific BC and transmission parameters DIFS, CWmin and CWmax, which can be considered as four instantiations of the DCF procedure. In addition, 802.11e introduces the Transmission Opportunity (TXOP) parameter (also per-queue), which supports that two stations can exchange a number of Data-ACK pairs with just a SIFS time between each frame, for up to TXOP units of time. Along the same lines, the amendment also introduces an optional unicast channel access policy, namely, the BA for reducing the channel access time overheads. With BA, instead of continuous Data-ACK exchanges, a transmitter sends a burst of data frames, separated by a SIFS interval, addressed to the same destination. Then, the transmitter polls the receiver with a BA Request (BAREQ) to trigger the transmission of the BA Reply (BAREP) frame, which contains the indication of the success/failure frames embedded into a bitmap. 802.11n amendment does not only make this functionality mandatory for High Throughput NICs (HT-PHY), but it also introduces the HT-ACK feature for the acknowledgment of Aggregated MPDU (AMPDU). This mechanism enables the transmission of many MPDUs with a single physical layer preamble and without spacing in-between. In this case, the destination, upon receiving the AMPDU, may transmit the BA right after a SIFS, and no BAREQ is required from the sender.

2.1.2 Multicast Video Transmission with legacy 802.11

The choice of multicasting or unicasting audio and video content over IEEE 802.11 networks does not depend only on the number of receivers, since the multicast transmission in the IEEE 802.11 standard has some speci cities that we consider in the following.

2.1.2.1 Multicast service

IEEE 802.11 de nes layer-2 multicast as the transmission of data frames with a multicast address as the Destination Address. This multicast address accounts for a set of stations. The basic access procedure corresponds to the DCF mechanism. In addition, no ACK is transmitted by any of the recipients of the frame. The lack of MAC-level recovery on multicast frames entails a reduced reliability of this kind of tra c, due to the increased probability of losing frames from interference or collisions. To that aim, the multicast frames must be transmitted at one of the rates included in the Basic Rate Set as these rates are more robust against errors. This set is de ned at the AP and includes the minimum set of rates that a station must support in order to join the Basic Service Set (BSS). The Basic Rate Set (BRS) usually includes rates with lower order modulations, so the transmission of multicast frames is performed at a reduced data rate, thereby the transmission occupies the channel longer time and this decreases the overall performance of the BSS.

2.1.2.2 Unicast service

The second choice for transmitting audio/video frames is the use of unicast tra c. Unlike multicast, unicast tra c can be transmitted at any rate and it is acknowledged and retried, so its reliability is higher than standard multicast tra c. The counterpart is the amount of bandwidth required to transmit video to multiple receivers as the number of ows grows with the number of stations receiving it, hence it is only suited for low bit rate ows and a reduced number of receiving stations.

2.1.2.3 Prioritization

With the introduction of 802.11e, multicast tra c can be di erentiated, so the network can be con gured to prioritize the multimedia ows while accessing the medium compared with the rest of tra c. Although this is an advantage, one of the problems with this approach is the de nition of only one queue for video tra c, as current video codecs generate video ows as a sequence of frames that relate to one another, with di erent levels of relevance at the decoding stage, hence multimedia tra c may bene t from intra-tra c category di erentiation.

2.1.3 802.11aa amendment: Robust Audio and Video Streaming for 802.11

As explained in the previous section, the availability of mechanisms to perform video streaming over 802.11 WLANs is rather limited. The IEEE 802.11aa amendment has been designed to specifically address multimedia transmission, implementing a set of new functionalities over the base specification. In this section, we describe the main novelties introduced by 802.11aa [10]. The objective of these extensions is to efficiently improve the reliability of audio and video streaming, while maintaining and even improving the service as perceived by the other streams. Following this, we consider the next mechanisms:[2] (*i*) Interworking with IEEE 802.1 Audio Video Bridging (AVB); (*ii*) Stream Classification Service (SCS); and (*iii*) GATS. Next we present the first two functionalities of the 802.11aa amendment, while the third one, which is the focus of our work, will be detailed stand-alone in Section 4.2.

2.1.3.1 Interworking with IEEE 802.1AVB

Within the 802.11aa standard the use of IEEE 802.1Qat [20], and specifically the Stream Reservation Protocol (SRP), is specified in order to enable end-to-end reservation of resources across IEEE 802 networks. Multimedia flows are the clear example of traffic that may take benefit of end-to-end resource reservation and as such it is envisioned to support and integrate the SRP.

It is important to note that the Stream Reservation Protocol is a *Higher Layer Protocol*. As such, the IEEE 802.11 system of a non-AP station is not able to interpret the SRP messages and its integration consists on the station sending the requests to the AP. In order to support end-to-end reservations on the wireless link, the AP has co-located a higher layer entity called Designated MSRP[3] Node (DMN) that is in charge of processing the signaling required by SRP and invoking the required 802.11 reservation primitives.

2.1.3.2 Stream Classification Service

This service is built upon the EDCA access mechanism, which is based on different Access Category (AC)s, and supports traffic differentiation by means of priorities between queues and heterogeneous configuration of the contention parameters (namely, Arbitrary Inter Frame Space (AIFS), CW and TXOP). The SCS, along with the intra-AC Traffic Stream prioritization, enables classification using layer 2 and/or layer 3 signaling and increases the granularity of the service differentiation already provided by the EDCA mechanism.

[2] The IEEE 802.11aa amendment also specifies mechanisms to manage Overlapping BSS (OBSS)s, but for the purpose of this work we only consider the case of a single WLAN.

[3] Multiple Stream Reservation Protocol (MSRP)

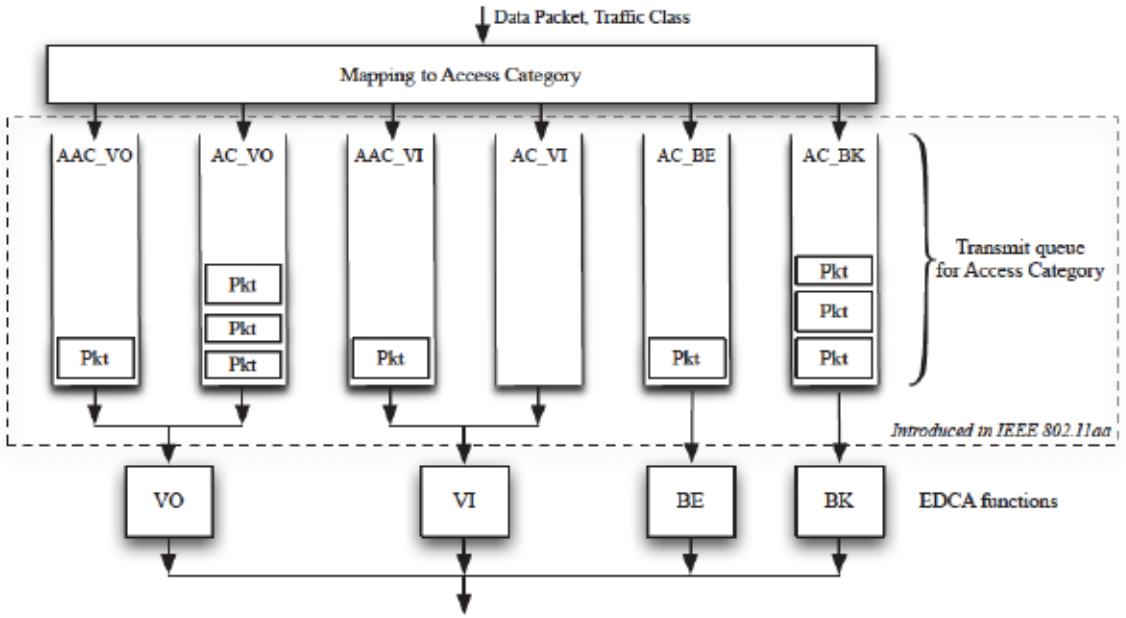


Figure 2.2: Stream Classification Service over the standard EDCA mechanism.

More specifically, as illustrated in Figure 2.2, this service introduces two additional queues within the existing EDCA access categories to support prioritization for audio and video streams, respectively. Furthermore, in addition to this intra-AC prioritization, packets are tagged with their drop eligibility (DE), which defines a different maximum number of (short and long) retries.

With this availability of additional access categories and the drop eligibility bit, *graceful degradation* of video quality is supported in case of bandwidth shortage. However, it should be noted that the SCS does not alter the standard MAC behavior, as it builds on the existing EDCA mechanism. Therefore, it inherits the same limitations of the EDCA mechanism with respect to video streaming, being either: (*i*) extremely unreliable, in case of legacy multicast, or (*ii*) extremely inefficient, in case of unicast transmissions to a moderate number of receivers (as we will see in Section 5.2).

2.2 Related Work

2.2.1 Voice Traffic in 802.11 WLANs

The standard operation of 802.11 introduces a large overhead for voice traffic, given the small size of its frames and its bi-directional nature. The payload of VoIP frames ranges between 60-160 bytes, while the mandatory headers that need to be added for their transmission is 52 bytes. The impact of such protocol overhead on VoIP has been extensively researched in the literature. The authors of [21] and [22] have investigated the

number of VoIP calls that can be supported in a WLAN with di erent 802.11 versions and di erent audio codecs. The degradation of voice performance in presence of low-priority data tra c has been analytically tackled in [7]. In that work, the authors propose an ACK skipping policy that optimizes the performance of voice frames. Other papers also discuss the importance of the MAC parameter settings on the voice performance, e.g., [23] and [24]. The scheme we propose achieves a much higher performance improvement than any proposal in the above papers.

The literature also provides simulation results and experimental studies based on COTS devices to measure the capacity of WLANs when voice tra c is present. For instance, the authors of [23] show that appropriate MAC tuning can improve capacity by 20% to 40%. Experiments reported in [25] con rm that commercial devices need non-trivial prioritization mechanisms in order to guarantee the quality of voice. Experiments in [22] show how voice conversations impair dramatically the performance of User Datagram Protocol (UDP) data tra c since they reduce the available bandwidth. All the above papers rely on the default MAC protocol operation; in contrast, in this thesis we implement a new mechanism at the driver and rmware level.

In this thesis we evaluate a new mechanism, namely VoIPiggy, designed to adapt the DCF protocol to the speci cs of voice tra c characteristics. The VoIPiggy mechanism can be related to the *reverse direction* (RD) mechanism of 802.11n [26], in which an AP *RD initiator* holding a TXOP, grants channel access to a station *RD responder* during the TXOP. The RD mechanism has been evaluated using simulations, for the case of voice and data tra c in [27], and for the case of online gaming applications in [28]. In contrast to these works, in this thesis we evaluate the performance of the proposed mechanism based on analytical and experimental results.

Besides, VoIPiggy presents similarities with the HCF Controlled Channel Access (HCCA) mechanism, in which the AP polls station for data. However, the use of HCCA has some shortcomings: (*i*) it introduces signi cant complexity, including, e.g., the scheduling con guration [29], (*ii*) it uses very small transmission rates for the data piggybacked on the Contention-Free (CF)-Poll frame, which results in the CF-Poll piggyback problem identi ed in [30], (*iii*) if the downlink voice frames are sent in the Contention-Free Period (CFP), additional downlink delay is incurred while waiting for this period [31], and (*iv*) if they are sent in the Contention Period (CP), the ACK for the uplink frame cannot be piggybacked, which leads to the use of three frames per exchange. In contrast, VoIPiggy achieves a substantially more e cient operation using a very simple scheme.[4] It is also worthwhile noting that HCCA has not been widely deployed, while here we present a working implementation of VoIPiggy.

In [32], authors present a novel framework for the rapid prototyping of new MAC

[4]For instance, a typical exchange at 48 Mb/s (data frames) and 12 Mb/s (control frames) for a 160-byte voice frame lasts 166 μs with VoIPiggy, while with HCCA in CP, it results in 227 μs.

schemes based on a programmable device, namely Wireless MAC Processor, which provides a set of MAC commands and a MAC engine to de ne and execute new MAC protocols. They show the use of piggybacking for Transport Control Protocol (TCP) acknowledgments as an example; their approach di ers substantially for ours, is based on a di erent implementation and is evaluated in a scenario with only two stations. Other former works have proposed the use of piggybacking to improve WLAN performance [33,34], yet their evaluation is performed exclusively via simulations.

The proposal that mostly relates to our work is *Softspeak* [35], which consists in aggregating voice frames at the AP, and a Time Division Multiple Access (TDMA)-like operation for voice stations based on *coarse-grained* time slots of 1 ms. These coarse slots not only limit the scalability of the proposal, but also require dealing with, e.g., slot allocation and time synchronization between nodes. Furthermore, the performance evaluation is carried out in a small scenario consisting of 10 nodes, and lacks analytical support.

2.2.2 Multicast Video Streaming Tra c in 802.11 WLANs

The video streaming to various receivers simultaneously over wireless networks is a challenging task addressed by multiple works in the literature. Most of the related work present in the literature focuses on nding new mechanisms or enhancements for the multicast transmission in IEEE 802.11 WLANs, motivated by the low tolerance to losses and the lack of retransmissions of the legacy scheme. Some of these works [36,37] propose the use of Forward Error Correction (FEC) codes, or multi-rate adaptation based on the feedback obtained from the stations.

A signi cant amount of work builds on the Leader Based Protocol (LBP). With LBP [8], one station is selected as the leader of the multicast group. This leader will send feedback to the AP with a positive ACK on behalf of the stations in the group, although they can send negative ACKs (NACK) to notify the reception of incorrect frames. An enhancement by [38] proposes the Enhanced Leader Based Protocol (ELBP), which is a hybrid between the Batch Mode Multicast MAC protocol (BMMM) [39] and LBP protocols, similar to the GCR solicited mechanism speci ed in 802.11aa, by using the Block Ack mechanism from 802.11e. In a follow-up work [40], the authors extend the previous work by proposing two mechanisms for leader selection. Authors in [41] advocate for the use of propagation-related metrics for leader selection (i.e., the station with worst link quality), supporting some results with experiments, although there are some practical issues due to synchronization and the capture e ect. In [42], the authors propose a combination of LBP with adaptive transmission rate, relying on the feedback from the stations and combined with a CTS-to-Self so that the multicast receivers will be aware of any pending transmission. Authors in [43] de ne a hybrid model that combines an LBP approach, adaptive rate transmission and tuned power control based on the positive and

negative feedback jamming ratio.

Apart from LBP-based proposals, in [44] authors propose to use *tones* to transmit the feedback information, i.e., NACKs or Negative CTS (NCTS), so the number of collisions gets reduced. However, this solution requires a dedicated signaling channel that is implemented with two wireless interfaces. Other proposed works mix schemes along the lines of Direct Multicast Sequence (DMS): in [45], authors propose to transmit the video I-frames using unicast, while the rest of video frames are transmitted using legacy multicast. Similarly to DMS, this proposal poses scalability issues.

Based on the above, most of the existing work has proposed non-standard schemes to improve video performance. Apart from the qualitative description in [46], only [47] presents some numerical gures on the performance of the recent 802.11aa standard. However, in contrast to our work, in that paper authors assume an 802.11b scenario with a low-bandwidth video, they only consider a subset of the existing 802.11aa mechanisms, and the performance evaluation does not address the e ciency of the schemes (in particular, for a minimum guarantee required).

Chapter 3

Voice Traffic in 802.11 Networks: VoIPiggy Mechanism

In this chapter, we propose the VoIPiggy mechanism, which piggybacks the voice frames onto MAC acknowledgments and enhances the voice performance in WLAN scenarios. First, we present the motivation behind the scheme and describe the mechanism as well as the modifications that it introduces to the standard operation of APs and stations. To this aim, we provide an analytical model to evaluate the performance in terms of: throughput, number of simultaneous supported voice calls and delay of voice traffic. Furthermore, we implement the proposed scheme and its functional modules. Finally, we present our testbed and report the extensive experimental evaluation conducted for the proposed algorithm in a wide set of network conditions and also with real voice applications.

3.1 Motivation

The standard operation of 802.11 introduces a large overhead for voice traffic, given the small size of its frames and its bi-directional nature. To quantify this overhead, let us consider a scenario consisting of one AP and one station, and the exchange of two voice frames between them, one per direction. Not including the impact of the backoff operation, the frame exchange follows the operation illustrated in the upper part of Figure 3.1. According to this, the total time required to perform this exchange following the standard operation is given by:

$$T_{std} = 2\left(DIFS + 2T_p + \frac{H + l_v}{R} + SIFS + \frac{ACK}{R_c}\right), \tag{3.1}$$

where $DIFS$ and $SIFS$ are constant times defined by the standard, T_p represents the duration of the preamble, l_v is the length of the voice frame, including the IP and UDP

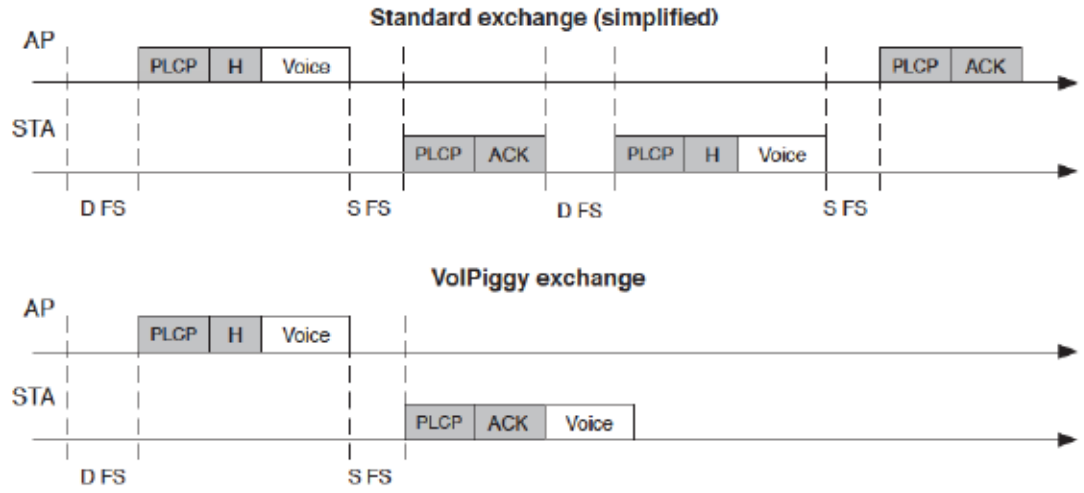


Figure 3.1: Frame exchange for the standard case (top) and our VoIPiggy mechanism (bottom).

Table 3.1: Efficiency of the standard for the case of the short-length frame exchange of Figure 3.1

Mode	R (Mb/s)	R_c (Mb/s)	T_{std} (μs)	$T_{\min}$ (μs)	η	$\frac{2T_{ack}}{T_{std}}$
802.11b	1	1	3084	1408	46%	20%
	2	2	1964	704	36%	26%
	5.5	2	1323	256	19%	39%
	11	2	1139	128	11%	45%
802.11g	6	6	553	235	42%	20%
	9	6	441	156	35%	25%
	12	6	385	117	30%	28%
	54	24	226	26	12%	36%

headers, H is the layer-2 header, ACK is the length of the acknowledgment, and R and R_c are the transmission rates for data and control traffic, respectively.

The minimum time required to perform the exchange is given by the time devoted to exchange voice data without any overhead, i.e., $T_{\min} = 2l_v/R$. Based on this, we can define the *efficiency* of the standard exchange as $\eta = T_{\min}/T_{std}$. In Table 3.1, we provide some values of this (in)efficiency for different combinations of the MCS of data and control frames, for 802.11b and 802.11g, assuming a frame length of $l_v = 88$ bytes, which results from adding the corresponding IP and UDP headers to a voice frame of 60 bytes.

The values obtained show that, for these typical MCSs, efficiency becomes worse as transmission rates increase, and reaches values as high as 46%. Note that we have considered the best conditions (i.e., only two stations, no backoff operation) and therefore this is the *best possible case* for the efficiency, which will be even smaller under more

realistic network conditions (e.g., higher number of stations or presence of hidden nodes that will increase the probability of collision and defer the stations access to the medium). Motivated by these poor gures, we next present the modi cations introduced by VoIPiggy to improve performance.

3.2 Description of the VoIPiggy mechanism

We identify the following sources of ine ciency in the standard exchange depicted in Figure 3.1: (*i*) after the reception of the voice frame from the AP in the downlink direction, the station has to send two consecutive frames in the uplink direction (i.e., from the station to the AP), namely an acknowledgment frame and a voice frame, separated by a $DIFS$ time, each of which involves a substantial overhead; (*ii*) furthermore, the second of these two frames is transmitted after contending for channel access, which adds an extra overhead; and (*iii*) the AP nally sends an acknowledgment frame back to the station to con rm the correct reception of the second voice frame, again involving an additional overhead. In order to eliminate these sources of ine ciency, our mechanism relies on the following key ideas:

The rst key idea of our proposal is to send the voice frame in the uplink direction after a $SIFS$ time following the transmission of the voice frame in the downlink direction. In this way, we take advantage of the fact that the medium is already cleared for transmission after the rst voice frame, and hence save the overhead due to a new contention.

The second idea is to merge the two consecutive frames in the uplink direction (the acknowledgment and the voice frames) into one single transmission by using the ACK to carry the second voice frame in the opposite direction. This saves the channel time devoted to the transmission of the ACK - given that the length of the MAC headers of the voice frame is relatively similar to that of an ACK frame - improves substantially the e ciency with low implementation costs.

The third idea is to omit the last ACK frame. This is based on the argument that the probability that a frame su ers from a failure in the uplink direction is much smaller than in downlink, as the uplink frame is sent after a SIFS, and hence is protected from collisions, which is the most common source for failures in 802.11. To ensure that uplink frames are not lost even in the unusual case that they su er failures, we have designed the mechanism described in Section 3.2.1, which retransmits failed uplink transmissions without requiring to acknowledge these frames.

Based on the above ideas, we propose to modify the operation of the protocol for the case of voice tra c as illustrated in the bottom part of Figure 3.1. Our proposal, as we

formally specify in Section 3.2.1, introduces two small changes to the standard operation of the station and the AP: (i) assuming that there is a pending voice frame in the output queue of the station (this is further discussed in Section 3.2.2), a $SIFS$ time after the reception of a voice frame from the AP, the standard ACK reply is replaced by a di erent frame, which includes both the control information corresponding to the ACK and the voice frame from the station; (ii) upon the reception of this new frame from the station, the AP proceeds as if a standard ACK was received (thus clearing the output queue) and it processes the voice data carried by the frame, but does not acknowledge the reception of the voice frame.

As Figure 3.1 illustrates, these modi cations save approximately the time spent in the transmission of the two ACK frames ($2T_{ack}$, where $T_{ack} = SIFS + T_p + ACK\ \ R_c$). According to the values in the last column of Table 3.1, this results in savings of between 20% and 45% over the standard exchange T_{std} (depending on the MCS used). Furthermore, an additional improvement is that, by means of this scheme, the channel contention is largely reduced as voice stations are polled by the AP, thus yielding additional e ciency improvements as detailed in the performance evaluation of Section 3.5.[1]

Note that the VoIPiggy design relies on the assumption that voice applications do not implement silence suppression. We argue that this is not a limiting assumption in realistic scenarios, since the most widely used voice applications nowadays, such as e.g. Skype or Google Hangouts,[2] do not implement silence suppression [48]. As reported in [49], this results in better voice quality and maintains UDP bindings at the Network Address Translation (NAT), among other advantages. In Section 3.5.5, we evaluate VoIPiggy using real voice applications (both Skype and Google Hangouts) and con rm experimentally that they do not use silence suppression.

3.2.1 Required Modi cations

Next, we formalize the operation of our proposal by describing the modi cations introduced in the standard operation of the AP and the stations.

Changes to the AP. Given that the mechanism is triggered by the voice frame transmitted by the AP, we give the highest priority to the delivery of this type of tra c by setting the contention window con guration for the voice queue to $CW^{VO}_{\min} = CW^{VO}_{\max} = 2$, which is the minimum allowed by the standard. This con guration has been chosen to ensure that the stringent delay requirements of voice tra c are satis ed.[3]

[1] The improvement on channel contention results from the fact that voice stations do not contend for the channel but piggyback their packets to ACK frames. Note that if we had designed VoIPiggy to piggyback the packets of the AP rather those of the stations, voice stations would contend for the channel and hence we would not see such improvement in channel contention.

[2] *Google Hangouts:* `http://www.google.com/+/learnmore/hangouts/`

[3] Even though we are prioritizing voice tra c, VoIPiggy not only improves the performance of voice but also of data tra c, as shown by the results of Section 3.5.

Another modi cation is the processing of the new piggybacked frames upon receiving them. To this aim, the AP has to identify the piggybacked frame. If the frame is received and the piggybacked data can be decoded, we proceed as if it were an ACK, removing the voice frame from the NIC queue and informing the upper layers, *without* triggering the transmission of an ACK. If the AP does not either receive the uplink frame carrying the ACK or cannot decode the voice data piggybacked to the frame, then it acts as it had not received the ACK and retransmits the downlink frame (following the standard backo procedure to retransmit frames).

Changes to the station. To take advantage of the savings introduced by the VoIPiggy exchange, the station needs to have a voice frame in the output queue, ready to be piggybacked. To this aim, we enforce that, when a station has a voice frame to transmit, it does not transmit it immediately but waits for a frame in the downlink direction. In order to avoid waiting too much time, which would harm voice performance, we limit the maximum waiting time by . The value of adapts to the voice codec used in order to minimize the delay su ered by outgoing frames, as we describe next.

Upon the reception of a voice frame from the AP, the station has to forge a frame of type `Data+ACK` and transmit it after a $SIFS$. In case of an uplink transmission failure, following the AP operation described above, the AP retransmits the downlink frame; the station identi es such a retransmission as it has the Retry ag in the header set, and replies by retransmitting the failed frame (piggybacked to an ACK).

With the aforementioned required modi cations, the operation of VoIPiggy resembles that of a *polling scheme*: the voice frames sent by the AP play the role of *poll* frames, to which the stations respond (when polled) by sending the voice frames in the opposite direction (to the AP). This similarity has already been mentioned in Chapter 2, where we report the main di erences between VoIPiggy and a polling scheme such as HCCA. In the rest of the chapter, we refer to this way of operating as *polling*-like operation.

We illustrate the changes introduced by VoIPiggy over the standard IEEE 802.11 state machine in Figure 3.2. Modi cations to standard operation of 802.11 are represented by means of shadowed circles and dashed or dotted lines, depending on whether they a ect the AP (dashed lines) or the station (dotted lines). A detailed description of the implementation of the above modi cations in our platform, performed at the driver and rmware levels, is provided in Section 3.4.

3.2.2 Setting of

In the following, we design the algorithm to compute . This parameter determines the maximum time a station holds a voice frame while waiting for a frame in the downlink direction. Following the explanation provided above, we want to be: (i) large enough to make sure that we piggyback as many frames as possible and (ii) as small as possible while meeting this condition, to avoid introducing unnecessary delays.

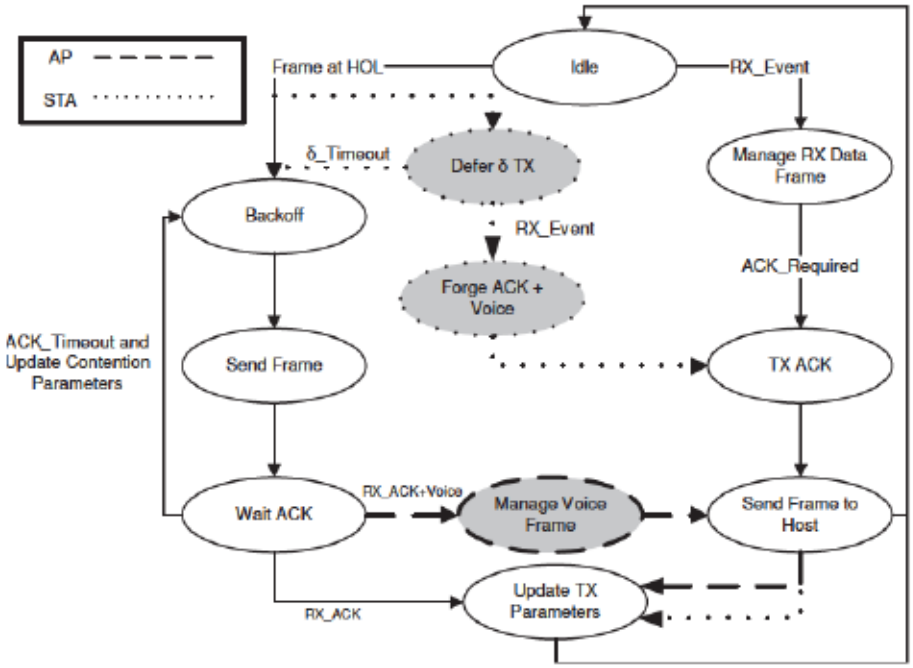


Figure 3.2: Changes introduced in the standard IEEE 802.11 MAC state machine by VoIPiggy. Dashed and dotted lines represent new state transitions, while shadowed circles represent new states.

Ideally, in voice traffic the inter-arrival time between two consecutive frames is fixed, equal to a constant term T (typically, $T = 20\,\text{ms}$). In this ideal case, it would be sufficient to set $\delta = T$ to make sure that the above objective is met since, with this setting, even in the very *worst case* that a voice frame is generated right after receiving a frame from the AP, it will be held long enough to wait until the next frame of the AP. However, in reality the spacing between frames from the AP may not be perfectly regular as there may be some deviations caused by, e.g., the backoff process. To account for these deviations, we follow a similar approach to the "Algorithm 4" proposed in [50].

Let t_i be the time we receive the i^{th} frame from the AP, and T_i be the estimation of the *average* inter-arrival time up to this frame. The computation of T_i is performed as:

$$T_i = (1 - \alpha)T_{i-1} + \alpha(t_i - t_{i-1}), \tag{3.2}$$

where α is a fixed constant (following [50], we set $\alpha = 0.125$). Furthermore, we also estimate the average deviation of the inter-arrival time from the estimated average as follows:

$$v_i = (1 - \alpha)v_{i-1} + \alpha|t_i - t_{i-1} - T_i| \tag{3.3}$$

Once we have calculated the above estimates, we set δ_i (the value of δ at the i^{th} frame) as:

$$\delta_i = T_i + Kv_i \tag{3.4}$$

where K is a positive constant (following [50], we set $K = 4$). The purpose of the Kv_i term is to set large enough to absorb the deviations su ered by the inter-arrival times. Note that, in case these deviations are negligible, then the term v_i will be very small and we will have $_i$ T, which corresponds to the ideal case mentioned above. The e ectiveness of the above algorithm for the setting of is experimentally evaluated in Section 3.5.

3.3 Performance Analysis

We next analyze the performance of the VoIPiggy mechanism presented in the previous section in terms of throughput performance, capacity region and delay of voice tra c.

3.3.1 Throughput performance

We consider a WLAN scenario with one AP and n_v VoIP associated clients, all of them with the VoIPiggy functionality enabled. Each VoIP client is connected to a remote client located outside the WLAN. We assume that the VoIP application generates packets of xed size l_v. In the same WLAN, we consider the presence of another set of n_d independent stations, each generating data frames of xed length l_d.

As a result of our scheme to con gure the parameter , we can assume safely that all frames from the n_v clients are piggybacked. Therefore, voice tra c is served by the AP in a *polling*-like way, with only one station (the AP) contending for the channel. Thus, we can model the voice activity in the WLAN as a single virtual station (see Figure 3.3). We denote with $_v$ the probability that this virtual station transmits in a *slot time* [51]. Similarly, we denote with $_d$ the probability that a data station transmits in a slot time.

Let $p_{s\,v}$ and $p_{s\,d}$ be the probabilities that a successful transmission occurs in a slot time for a VoIP station and a data station, respectively, and let T_v and T_d be the corresponding slot lengths in these cases, which can be computed as:

$$T_v = DIFS + SIFS + 2T_p + \frac{H + ACK\ + 2l_v}{R} \tag{3.5}$$

$$T_d = DIFS + 2T_p + \frac{H + l_d}{R} + SIFS + \frac{ACK}{R_C} \tag{3.6}$$

where ACK is the length of the modi ed acknowledgment header for piggybacked voice packets. Throughout the article we will assume that the parameters frame lengths, MCS are such that $T_d > T_v$, which is the typical case, although the model could be easily extended to account for di erent settings of the parameters (following [9]).

Similarly, we denote with $p_{c\,d}$ and $p_{c\,dv}$ the probabilities of having a collision involving data stations only and voice and data stations, respectively (note that under our assumptions voice stations do not collide with each other), and with $T_{c\,v}$ and $T_{c\,dv}$ the

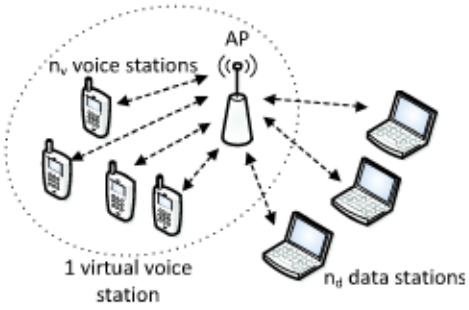


Figure 3.3: Scenario: WLAN with n_v voice stations and n_d data stations.

corresponding duration of these collisions. Finally, let p_e be the probability of having an empty slot (the duration of such a slot time is a constant specified by the standard, T_e). With this notation, the average duration of a slot time can be expressed as:

$$T_{slot} = p_e T_e + p_{c,d} T_{c,d} + p_{c,dv} T_{c,dv} + p_{s,v} T_v + p_{s,d} T_d. \tag{3.7}$$

With the above, the throughput of a voice and a data station in the WLAN (denoted as R_v and R_d, respectively) can be computed as:

$$R_v = \frac{1}{n_v} \frac{p_{s,v}\, l_v}{T_{slot}}, \quad R_d = \frac{1}{n_d} \frac{p_{s,d}\, l_d}{T_{slot}}. \tag{3.8}$$

Assuming that all data stations and the virtual one are independent, the probabilities above can be computed as follows:

$$p_e = (1-\tau_v)(1-\tau_d)^{n_d}, \tag{3.9}$$

$$p_{c,dv} = \tau_v \left(1-(1-\tau_d)^{n_d}\right), \tag{3.10}$$

$$p_{c,d} = (1-\tau_v)\left(1-(1-\tau_d)^{n_d} - n_d \tau_d (1-\tau_d)^{n_d-1}\right), \tag{3.11}$$

$$p_{s,v} = \tau_v (1-\tau_d)^{n_d}, \tag{3.12}$$

$$p_{s,d} = n_d \tau_d (1-\tau_d)^{n_d-1}(1-\tau_v). \tag{3.13}$$

Finally, to compute the slot time durations, we use T_d and T_v as defined in (3.6) and (3.5), and we consider that the collision duration is determined by the longest frame exchange, which leads to:

$$T_{c,dv} = \max(T_v, T_d) = T_{c,d} = T_d. \tag{3.14}$$

With the above, we can compute the throughput obtained by each traffic type (voice and data) for a given scenario of n_v and n_d stations *if the probabilities τ_d and τ_v are known*. Hence, the remaining challenge is to determine the value of these probabilities, which we refer to as the *point of operation*. We next address this challenge.

3.3.2 Point of operation

Here, we compute the pair (τ_d, τ_v) given the transmission parameters of the WLAN, the input variables n_v, n_d, l_v, l_d and the traffic generation rates of a voice flow (r_v) and a data flow (r_d).

Our analysis follows the technique described in [9]. We define *saturation rate* as the rate that a station would obtain if it always had a packet ready for transmission (i.e., if it were constantly backlogged). Based on this definition, we can classify stations as *saturated* (when the traffic generation rate is above the saturation rate) or *non-saturated* (when the traffic generation rate is below the saturation rate). We first describe how to compute the transmission probabilities depending on whether the traffic type is saturated or not, and then address the general case.

Saturated stations. We first consider the case in which stations are saturated. Following [51], the transmission probability of a station can be computed based on its minimum contention window W, and its conditional collision probability p. For the case of voice traffic, given their backoff configuration, they always use the same contention window independent of the number of retransmissions, i.e., $W_v = CW_{min}^{VO} = CW_{max}^{VO}$, and thus we obtain the following result:

$$\tau_v = \frac{2}{1 + W_v}. \tag{3.15}$$

For the case of data traffic, the conditional collision probability can be computed as the probability that at least another station transmits data or voice:

$$p_d = 1 - (1 - \tau_d)^{n_d - 1} (1 - \tau_v); \tag{3.16}$$

the corresponding transmission probability is [51]:

$$\tau_d = \frac{2}{1 + W_d + W_d p_d \sum_{i=0}^{m-1} (2 p_d)^i}, \tag{3.17}$$

where m is the maximum backoff stage and W_d is the CW_{min} used for the data traffic type.

The system (3.15)–(3.17) can be solved numerically to obtain the transmission probabilities (τ_d, τ_v), and based on these we can use (3.8) to compute the voice and data throughput under saturation conditions.

Non-saturated stations. Under non-saturation, we assume that all traffic generated is served, and therefore the following holds for the case of the AP (i.e., voice traffic):

$$\frac{1}{n_v} \frac{\tau_v (1 - p_v)\, l_v}{T_{slot}} = r_v, \tag{3.18}$$

where p_v is the conditional collision probability for voice, given by the probability that at least one data station transmits:

$$p_v = 1 \quad (1 \quad _d)^{n_d} \tag{3.19}$$

Similarly, for the case of a data station, we have:

$$\frac{_d(1 \quad p_d)\, l_d}{T_{slot}} = r_d \tag{3.20}$$

where p_d is computed as in Eq. (3.16).

Therefore, once n_d, n_v, r_v and r_d are known, and assuming that both types of tra c are not saturated, $_d$ and $_v$ can be obtained by solving the system of equations (3.16), (3.18) (3.20).

General case. We next combine the above analyzes for the saturated and non-saturated cases to obtain the operational point of the WLAN in a generic scenario in which, depending on the o ered load r_v and r_d, neither, only one or both tra c types are saturated. In order to compute the point of operation, we proceed iteratively as follows.

1. We rst assume that all stations are saturated and compute $_v$ and $_d$ by solving the system (3.15) (3.17).

2. Using the transmission probabilities obtained, we compute the per-station throughput for each type of tra c, R_v and R_d, with (3.8).

3. If the obtained throughput is less than or equal to the tra c generation rate for both data and voice tra c, the analysis is terminated and the pair ($_v$ $_d$) determines the point of operation of the WLAN. Otherwise, there are three possible choices:

   (a) Neither voice tra c nor data tra c are saturated (i.e., $R_v > r_v$ and $R_d > r_d$). In this case, the system of equations to solve is given by (3.16), and (3.18) (3.20).

   (b) Voice tra c is not saturated and data tra c is saturated (i.e., $R_v > r_v$ and $R_d < r_d$). In this case, the system of equations to solve is given by (3.16) (3.19).

   (c) Voice tra c is saturated and data tra c is not saturated (i.e., $R_v < r_v$ and $R_d > r_d$). In this case, the system of equations to solve is given by (3.15), (3.16), and (3.20).

4. We next solve the corresponding system of equations to obtain the probabilities $_v$ and $_d$, computing again the throughputs R_v and R_d. If these throughputs meet the same conditions as the ones used to derive the corresponding $_v$ and $_d$ probabilities, the algorithm terminates. Otherwise, we go back to step 3.

3.3.3 Capacity region

Based on the above analysis of the point of operation, we now address the issue of computing the set of feasible allocations in a WLAN, i.e., the number of flows that can be supported without throughput losses, which we refer to as the "capacity region" of the WLAN. Our capacity region analysis assumes that all voice stations use the same codec, and that all data stations generate the same traffic, although these assumptions could be easily relaxed at the cost of a more complex derivation.

Voice-only scenarios. We first consider the simple case when there is only voice traffic in the WLAN. As explained above, we assume that in this case VoIPiggy enforces a *polling*-like operation across voice stations. In these circumstances, all voice traffic can be served as long as the total arrival rate at the AP is below the maximum service rate R_v^*, which is computed from Eqs. (3.7)–(3.13) with $n_v=1$ and $n_d=0$:

$$R_v^* = \frac{\tau_v l_v}{(1-\tau_v)T_e + \tau_v T_v}, \tag{3.21}$$

where τ_v is computed via Eq. (3.15), i.e., in saturation conditions. Assuming a generation rate of r_v, the maximum number of conversations in a voice-only scenario, which we denote as n_v^*, can then be computed as:

$$n_v^* = \left\lfloor \frac{R_v^*}{r_v} \right\rfloor. \tag{3.22}$$

Voice and data scenarios. We next consider the general scenario in which there are voice and data stations. We say that a given set (n_v, n_d) of voice and data stations is supported by the WLAN if all the traffic generated by the stations can be sent to the WLAN, i.e., both traffic types are not saturated and hence there are no losses. In this case, we say the pair (n_v, n_d) lies within the capacity region $\mathcal{C}(r_v, r_d)$ of the WLAN, i.e.,

$$(R_v = r_v, R_d = r_d) \Leftrightarrow (n_v, n_d) \in \mathcal{C}(r_v, r_d)$$

Based on our analytical model, one way to compute the convex hull of $\mathcal{C}$ is by performing a sweep on n_v from 0 to n_v^*, and for each value of n_v increase the number of data stations n_d until any of the two traffic types (voice or data) becomes saturated.[4]

3.3.4 Delay performance

The above analysis serves to determine if a set of voice and data flows is supported by the WLAN; however, in order to provide QoS guarantees to voice applications, more

[4] Although we have considered that data traffic is generated at a finite rate r_d, following a similar procedure we could compute, e.g., the maximum rate that one data station could get in presence of n_v voice flows.

sophisticated models involving delay performance are required. To tackle this, we next analyze the delay performance of voice in a WLAN under the VoIPiggy mechanism. Our analysis focuses on the downlink direction, i.e., traffic from the AP to the stations, as this is the "bottleneck" of the network (in the uplink direction, stations hold their frames up to δ, and hence the delay is bounded by a low value).

Voice-only scenario. We first analyze the delay performance of a WLAN serving n_v flows. Given that the minimum value of CW is used and there is no data traffic present, we can assume that service time for voice frames is mostly constant and equal to T_v. In these conditions, our system is a single server queue with periodic arrival processes and deterministic service times, and can thus be modeled following [52]. With this model, the survivor function $F^{-1}(t)$ for the queuing delay in this scenario can be computed as:

$$F^{-1}(t) = \frac{P_{n_v-1}(t, t_v, T_v)}{t_v^{n_v-1}} \quad , \quad x \geq 0, \tag{3.23}$$

where t_v is the inter-arrival time of voice traffic, given by $t_v = l_v/r_v$, and $P_k(t, t_v, T_v)$ is computed as:

$$P_k(t, t_v, T_v) = \sum_{l=0}^{k-1} q_{k,l}(t, T_v)(t_v - kT_v + t)^l, \tag{3.24}$$

with the coefficients $q_{k,l}(t, T_v)$ computed recursively as follows:

$$q_{0,l}(t, T_v) = 0, \tag{3.25}$$

$$q_{k,0}(t, T_v) = \left((kT_v - t)^+\right)^k, \tag{3.26}$$

$$q_{k,l}(t, T_v) = \frac{k}{l} \sum_{j=l-1}^{k-2} \binom{j}{l-1} T_v^{j-l+1} q_{k-1,j}(t, T_v), \tag{3.27}$$

where $y^+ = \max(0, y)$. From the above, the Cumulative Distribution Function (CDF) of the queuing delay can be obtained as $F(t) = 1 - F^{-1}(t)$.

The above provides the queuing delay suffered by a frame. To obtain the total delay, we need to add the transmission time of a voice frame from the AP to the station, which is given by:

$$T_v' = DIFS + T_p + \frac{H + l_v}{R}. \tag{3.28}$$

Note that, in contrast to T_v, the above transmission time only accounts for the time elapsed until the voice frame reaches the AP and hence includes neither the ACK nor the piggybacked voice frame. The addition of the queuing delay and transmission time leads to the following CDF for the total delay D of the delivery of downlink voice frames, which is a *shifted* version of $F(t)$:

$$F_D(t) = F(t - T_v'), \quad t \geq T_v'. \tag{3.29}$$

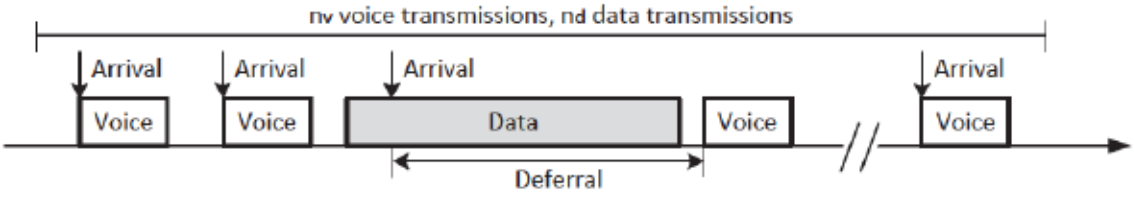


Figure 3.4: Channel deferral by the AP in presence of data traffic.

Once we have obtained the CDF for the total delay, we can obtain its numerical derivative $f_D(t)$, and from this value it is straightforward to obtain any performance figure for the downlink traffic in the voice-only scenario, e.g., average delay, 95-percentile or standard deviation.

Voice and data scenario. We next focus on the case when data traffic is also present in the WLAN. In this case, the impact of data traffic is twofold: voice transmission from the AP may collide with data transmissions and also the AP has to perform channel deferral whenever it senses the medium and detects that it is busy.

In [53], it has been observed that channel deferral dominates the performance in 802.11 wireless networks, while collisions have a much smaller impact. With VoIPiggy, the impact of collisions is even smaller, since there are fewer flows contending for channel access. Based on this, in our analysis we neglect the impact of collisions and assume that a voice frame from the AP is either transmitted immediately, or it has to wait until an ongoing data transmission has finalized (i.e., for a residual time), as shown by Figure 3.4. By denoting the probability of deferral by p_{def}, the CDF of the total delay can be expressed as:

$$F_D(t) = (1 - p_{\text{def}})U(t - T_v) + p_{\text{def}}F_D^R(t - T_v), \tag{3.30}$$

where $U()$ is the unit step function, and $F_D^R(t)$ is the CDF of the residual time of a data transmission. Given that these are of fixed length, if we assume that the residual time follows a uniform distribution, we can express $F_D^R(t)$ as:

$$F_D^R(t) = F_{\mathcal{U}}(t, T_d + t), \tag{3.31}$$

where $F_{\mathcal{U}}(0, T_d)$ is the CDF of a random variable uniformly distributed between 0 and T_d.

The pending challenge is to compute p_{def}. Our key approximation to compute this is to assume that whenever a voice frame is preceded by a data frame, it has to wait for the data transmission to end, as depicted in Figure 3.4. Thus, the probability of channel deferral is equal to the probability that the previous transmission corresponds to a data frame. Given that on average in a given period of time T there are n_v voice transmissions and n_d data transmissions, if we consider a tagged voice transmission out of the n_v, this

probability can be expressed as:

$$p_{\text{def}} = \frac{n_d}{n_d + n_v \quad 1} \tag{3.32}$$

which completes the model for the delay performance in a scenario with concurrent voice calls and data stations.

3.4 Implementation Details

In this section we detail the implementation of VoIPiggy, using Alix 2d2 devices from PC Engine.[5] We rst provide an overview of the chosen platform, and then describe the required modi cations to the Linux kernel and to the device rmware.

3.4.1 Platform overview

The Alix 2d2 embedded devices are popular low-cost computers, equipped with a Geode LX800 AMD 500 MHz CPU, 256 MB DDR DRAM, 2 Mini-PCI sockets and a Compact Flash socket, to which we attached a 4 GB card to accommodate the installation of a Linux distribution. We installed a Broadcom BCM94318MPG 802.11b/g MiniPCI wireless card, and use the Ubuntu 9.10 Linux (kernel 2.6.29) software platform.

The implementation of the 802.11 stack for the chosen platform is composed of three main software/ rmware modules, illustrated in Figure 3.5: (*i*) the `mac80211` framework that takes care of the high-layer operations; (*ii*) the device `Driver`, which is a wrapper between the internal bu ers and the physical device; and (*iii*) the `Firmware` of the device, which implements the internal logic that controls time-critical operations such as, e.g., packet retransmissions, which can be performed at neither kernel nor application level, due to the unpredictable delay introduced by the buses when crossing the protocol stack.

Linux kernel supports the Broadcom chipset using the `b43` open source driver, which loads the rmware during startup. We take advantage of this feature to substitute the default rmware with `OpenFWWF`,[6] an Open source FirmWare for WiFi networks that supports customization of the device internal operations and has been used in the past to introduce extensions to the 802.11 default behavior [54, 55]. The use of this rmware enables modifying the protocol state machine by reacting to some conditions, and grants access to key internal modules of the device, namely: (*i*) *MAC processor (MP)*, engine responsible for running the state machine; (*ii*) *Direct Memory Access (DMA) Controller*, which controls the data tra c from/to the host and interfaces the MP to the host kernel; (*iii*) *TX FIFO queues*, driven by the DMA controller, which deliver outgoing packets

[5] *PC Engines*: `http://www.pcengines.ch/`

[6] We refer the interested reader on the details of the `OpenFWWF` rmware to its web-page: `http://www.ing.unibs.it/openfwwf/`

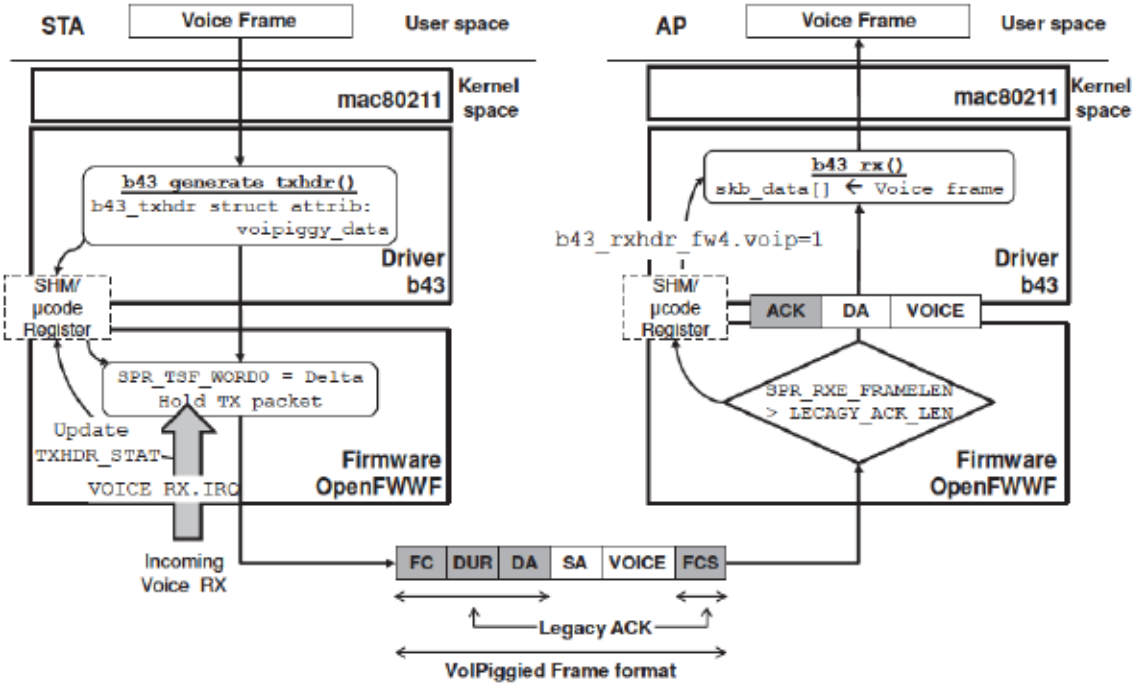


Figure 3.5: Implementation of VoIPiggy.

composed by the host kernel; (*iv*) *TX Engine (TXE)* that prepares a frame for transmission and waits for a transmission opportunity; (*v*) *RX Engine (RXE)* that decodes the signal received from the air, checks the CRC validity and reports the received packet length; (*vi*) *SHared Memory (SHM)*, where the MP maintains the state variables and may copy portion of TX and RX packets for inspection.

For the communication between the firmware and driver, we rely on the block of data that the driver attaches to the structure that contains each packet to enable per-packet configurations, according to decisions made by `mac80211`. We extend it to pass information to the firmware. In the following, we describe the required modifications to the device firmware and to the Linux kernel.

3.4.2 Firmware Modifications

Some of the functions of the VoIPiggy mechanism, being very time critical, need to be implemented at the firmware level. To this end we build our system on `OpenFWWF`.

In order to support VoIPiggy at the **station** side, we have to first hold outgoing voice frames, which are marked by the driver as we describe next, up to a maximum time δ. This is done by loading (upon the arrival of a voice frame) the time synchronization register `SPR_TSF_WORD0` with the δ value computed by the driver, and holding the frame until the register reaches zero or a voice frame arrives. Next, we have to modify the interrupt request triggered upon the reception of a frame. More specifically, we implement a new

procedure, VOICE_RX.IRQ, which is triggered when a voice frame from the AP arrives and the head-of-line frame is a voice frame. This procedure forges the VoIPiggy frame, which extends the default ACK to include the voice frame and the sender MAC addresses (updating the packet length and PLCP accordingly). After transmitting the voice frame, we continue to hold it in case it has to be retransmitted. This happens when a voice frame with the retry ag set is received from the AP. The frame is held until the AP transmits a new frame (with the retry ag unset) or the application generates a new voice packet.

At the **access point**, the main modi cation at the rmware level is the ability to identify and process the *voipiggied* frames received from the stations. For simplicity, we do this by comparing the length of the received frame, which is provided in the SPR_RXE_FRAMELEN register, against the length of a legacy ACK. If longer, and the piggybacked data is correctly decoded, the complete frame is passed to the driver for processing, as we describe in the next section; otherwise, the default ARQ procedure is triggered.

3.4.3 Driver Modi cations

The less time critical functions were implemented at the driver level in the kernel, in the C programming language.

At the **station** side, when the driver identi es a frame as VoIP,[7] it marks it as suitable for piggybacking. This is done by modifying the b43_generate_txhdr() function of the driver to extend the b43_txhdr data structure, in order to include a bit that we denote as PIGGYBACK_ENABLED. We also pre-compute in this function some variables (e.g., PLCP header) to speed-up the forging of the piggybacked frame in the rmware.

At the **access point** side, we add a *hook* in the driver to process the long ACK frames placed in the skb_buff bu er. This processing consists of rst forging a regular data frame for the voice payload, by inserting the missing sender and receiver addresses, the LLC header, etc. Then, the frame is passed to the mac80211 module (which remains oblivious to the VoIPiggy operation) via the b43_rx() function.

Finally, to compute the parameter, and also for debugging and monitoring purposes, we extended the b43_wldev structure that contains various transmission and reception statistics (e.g., total number of frames, transmission attempts) to include those related to the operation of VoIPiggy (e.g., inter-arrival time, number of *voipiggied* frames).

3.5 Performance Evaluation

In this section we rst describe the testbed used in our experiments. Then, we evaluate the performance of VoIPiggy under a large number of di erent scenarios, in terms of

[7] There are various alternatives to detect if the frame is VoIP, such as: by having the application set the *Di erentiated Services Code Point* (DSCP) byte, by looking at the *Payload Type* eld from a RTP header, at the frame length, etc. For simplicity we do it by checking if the destination port matches with a prede ned value.

Figure 3.6: Deployed testbed, consisting on 1 AP and 30 stations.

number of stations, data tra c and MCS used. Throughout our experiments we compare (whenever possible) the resulting performance against that obtained with the standard channel access, in which voice tra c and data tra c are con gured with the recommended CW parameters of the VO and BK queues of the EDCA standard, respectively (denoted as EDCA). Finally, we validate the algorithm used to compute , test it using real voice applications and provide additional insights on the performance of VoIPiggy.

3.5.1 Testbed Description

Our testbed deployed in an o ce space at University Carlos III de Madrid, Spain, is depicted in Figure 3.6. It consists of 30 Alix 2d2 devices acting as wireless stations, and one desktop machine acting as AP. The AP, in the lower left corner of the gure, uses a 7 dBi omnidirectional antenna located in the center of the testbed. The stations are deployed around this antenna and equipped with 2 dBi omnidirectional antennae. All nodes use a transmission power of 20 dBm. Given that the 2.4 GHz band is well populated in our testbed, we took great care in performing the experiments when the tra c activity was low.

We use `mgen`[8] as tra c generator, to emulate the behavior of standard voice codecs and data transfers. This tool supports the computation of one-way delay gures by inserting timestamps in all packets. Since this requires that nodes are synchronized, we run the *Precision Time Protocol (PTP) daemon*[9] over the Ethernet interfaces of the nodes, achieving time synchronization with 0.1 ms of accuracy. The wired interface is also used to perform other control and management plane operations, such as remote execution of tests or retrieval of the results for o -line processing, so that they do not interfere with the actual measurement data on the wireless medium.

3.5.2 Capacity Region

Voice-only scenarios. We start our performance evaluation with a scenario in which

[8] *MGEN*: `http://cs.itd.nrl.navy.mil/work/mgen/`
[9] PTP: `http://ptpd.sourceforge.net/`

Table 3.2: Maximum number of conversations supported in a voice-only scenario.

Voice codec	MCS (Mb/s)	n_v experimental EDCA	VoIPiggy	Gain	n_v model VoIPiggy	Gain
G. 711	2	5	9	80%	9	80%
	5.5	10	18	80%	18	80%
	11	12	26	116%	26	116%
	6	19	29	53%	29	53%
	9	24	30	-	49	71%
	12	26	30	-	52	100%
	54	30	30	-	125	-
G. 726	2	8	14	75%	14	75%
	5.5	10	26	160%	26	160%
	11	12	30	-	33	175%
	6	23	30	-	49	113%
	9	26	30	-	66	153%
	12	30	30	-	79	-
	54	30	30	-	154	-

only voice tra c is present. To this aim, we analyze the maximum number of conversations supported in the WLAN, with di erent con gurations of the voice codec and MCS used. More speci cally, we increase the number of stations n_v from 1 until $n_v + 1$ is reached, which is the minimum number of voice stations that leads to tra c losses. In order to maintain good statistical guarantees, for each con guration of n_v we repeat the test 5 times with a duration of 30 s each, increasing the number of conversations if none of the experiments su ers from tra c losses. We also obtain, following a similar procedure, the maximum number of voice conversations that can be supported with the standard EDCA con guration. We present the obtained results in Table 3.2, in which we also provide the value predicted by our analytical model of Section 3.3.3, as well as the resulting improvements in the maximum number of conversations of VoIPiggy over EDCA (columns Gain). Note that we mark those experiments in which the number of supported conversations reached the maximum number of nodes in our testbed as 30.

There are two main conclusions that can be drawn from the table. First, the experimental results show a perfect match with those from the theoretical model for all the cases that can be evaluated with the number of nodes available in the testbed (i.e., $n_v < 30$). Second, VoIPiggy signi cantly increases the e ciency of WLANs in the presence of voice tra c. Indeed, as compared to the standard recommended con guration, the improvements range between 53% and 175% - a dramatic gain at the relatively small cost of implementation.

The above con rms that, in voice-only scenarios, VoIPiggy is able to boost the capacity of the network, consequently reducing the ine ciency of the standard protocol when short packets are sent in both directions (as discussed in Section 3.1). We next analyze how

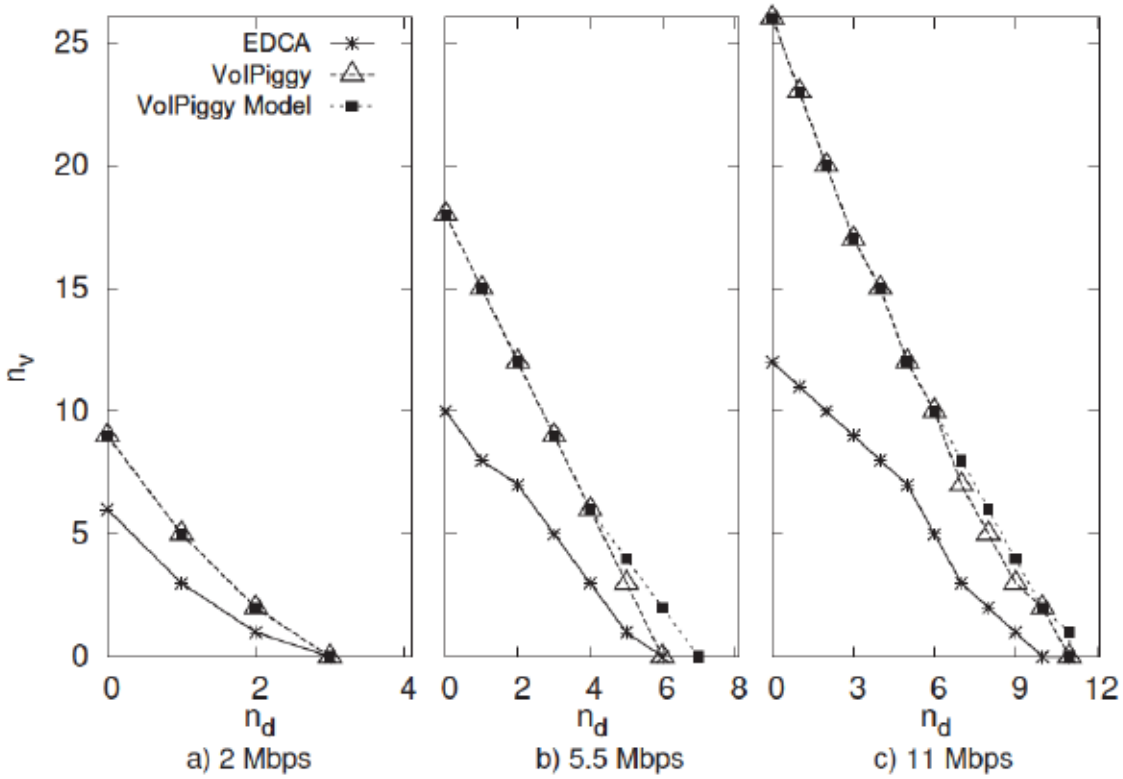


Figure 3.7: Capacity region of an 802.11b scenario with voice and data stations.

the efficiency improvements introduced by VoIPiggy in the delivery of voice also increase the capacity of WLANs when data traffic is present.

Voice and data scenarios. We next analyze the case of a scenario with n_v bi-directional voice conversations and n_d uplink data flows, in order to experimentally assess the capacity region $\mathcal{C}$ of VoIPiggy and to validate our analytical model presented in Section 3.3.3. To this aim, we perform a sweep on the number of voice stations n_v (from 0 to n_v^*) and, for each value of n_v, we sweep on the number of data stations n_d, starting from 1 and increasing it while the throughput demands of voice and data flows are satisfied. Like in the previous case, we repeat each test 5 times, and only increase the number of stations if no significant losses were measured.[10]

We start our measurements for the case of 802.11b, assuming the G. 711 voice codec and data stations transmitting at $r_d = 500$ kb/s, and the following values for the MCS=$\{2, 5.5, 11\}$ Mb/s. For each configuration of the transmission rate, we experimentally measure the capacity region for the EDCA configuration and for VoIPiggy, also providing the theoretical values in this case ('VoIPiggy Model'). The results are depicted in Figure 3.7. Experiments show that when there are no voice conversations in the WLAN there is no increase in the capacity, given that no piggybacking is taking place. In contrast, the more

[10]The capacity region figures are presented for the case when losses are below 1%. We have repeated the experiments for up to 10% losses, and have obtained very similar results.

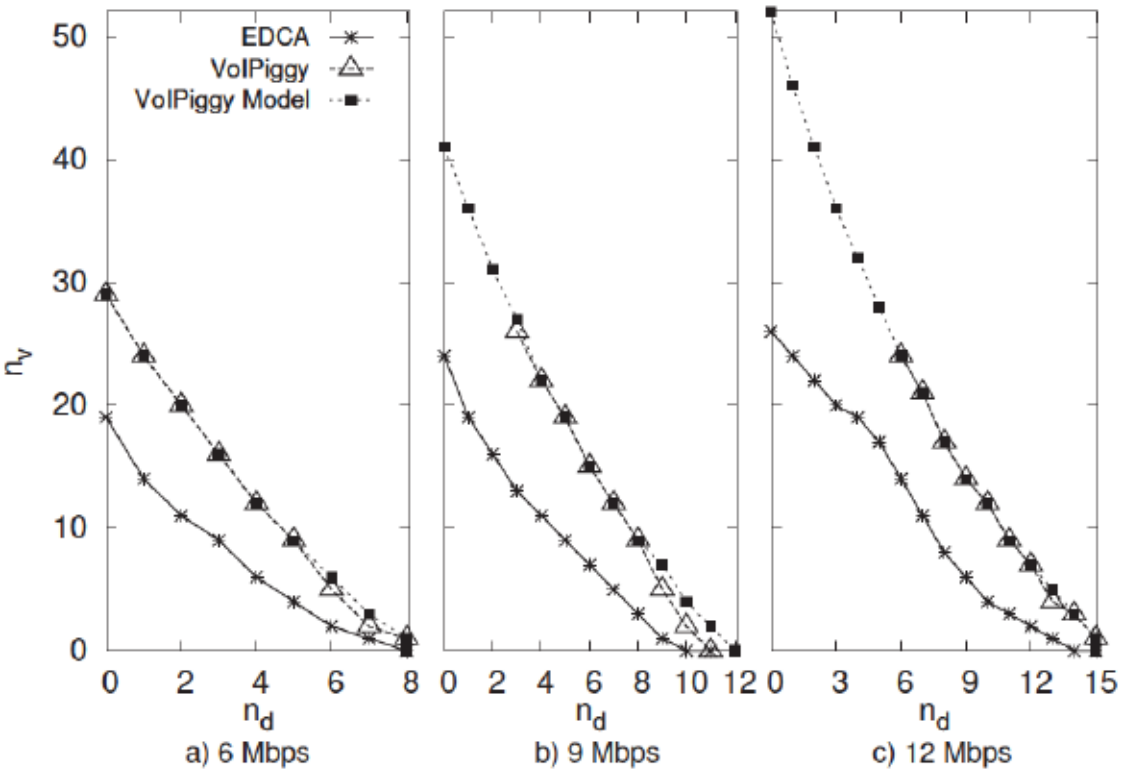


Figure 3.8: Capacity region of an 802.11g scenario with voice and data stations.

voice flows that are present, the larger the difference between the EDCA performance and the one achieved by means of VoIPiggy. Indeed, for the MCS presented in the figure, the "area" of the capacity region is practically doubled with the use of VoIPiggy. It is worth remarking the good match between experimental and analytical results.

We proceed similarly for the case of 802.11g, with the 6, 9, and 12 Mb/s MCSs. The results are depicted in Figure 3.8. Given that the MCSs considered are more efficient when dealing with voice traffic, the performance improvements are slightly below those presented in the previous case, but are still quite significant. As the capacity region $\mathcal{C}$ is notably large in this case, we can only confirm the accuracy of the model when $n_v + n_d \leq 30$. We also show in the figure the analytical results for topologies larger than our testbed, which helps to properly illustrate the large improvements due to VoIPiggy.

Based on these results, we conclude that VoIPiggy is able to almost double the capacity region of 802.11 WLANs, i.e., the number of stations that can be supported with no throughput loss, as confirmed both by analytical and experimental results. We next analyze the delay performance when using VoIPiggy for scenarios at the boundary of the capacity region.

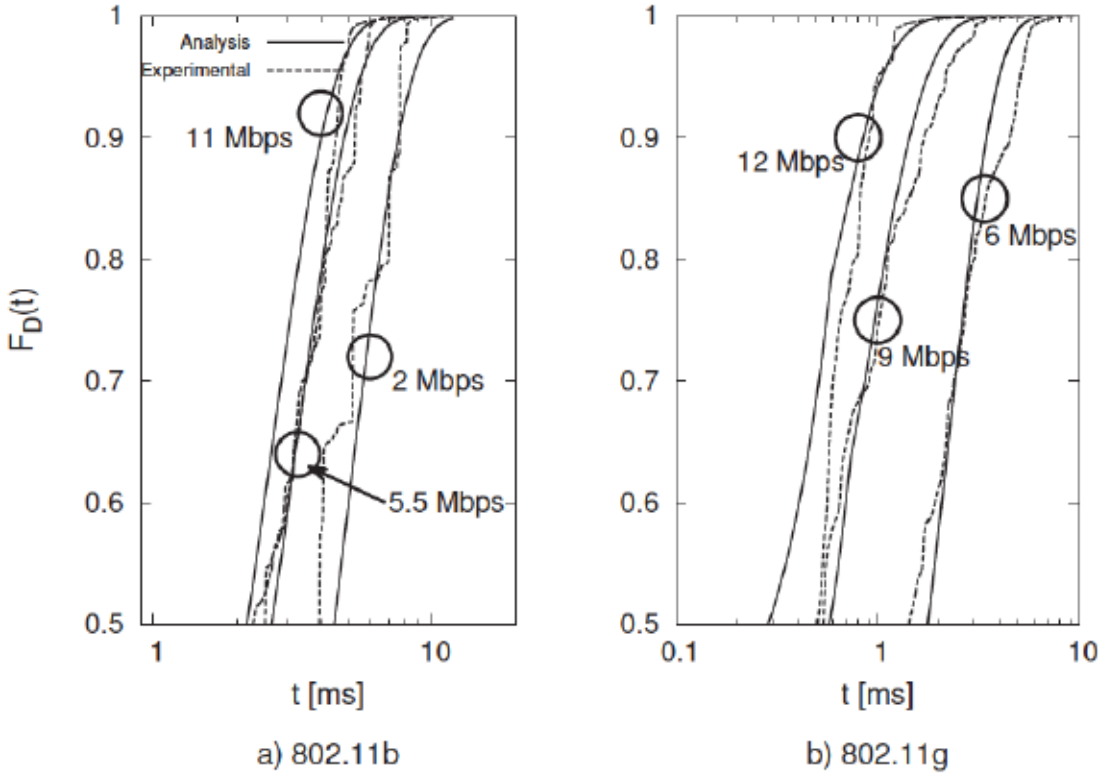


Figure 3.9: CDF of the delay for voice-only scenarios.

3.5.3 Delay Performance

The results of the previous section validate the performance improvements of VoIPiggy and the accuracy of the analytical model for the capacity region. Such results can be used, e.g., to perform call admission control (CAC) decisions if the performance metric of interest is throughput. Conversely, if we are interested in the *quality* of the service received by voice conversations, it is important to characterize delay performance in addition to throughput. To this end, we next analyze the CDF of the downlink delay using VoIPiggy.

Voice-only scenarios. We first consider the case of a voice-only scenario. We consider a scenario with the maximum number of conversations supported with VoIPiggy for different MCSs (i.e., n_v^*), all of them using the G.711 voice codec, and compute the CDF of the service delay. Each experiment is run for 60 s, repeated 5 times to confirm that results did not vary much across experiments. The results are depicted in Figure 3.9 for the case of 802.11b (left) and 802.11g (right). We use dotted lines to represent the experimental results, and solid lines for the numerical figures computed using the analytical model presented in Section 3.3.4.

We observe a good match between analytical and experimental values. Furthermore, results confirm the good service provided to voice traffic. For the case of 802.11b it only reaches 10 ms for the lowest MCS, while for the case of 802.11g delay figures are well

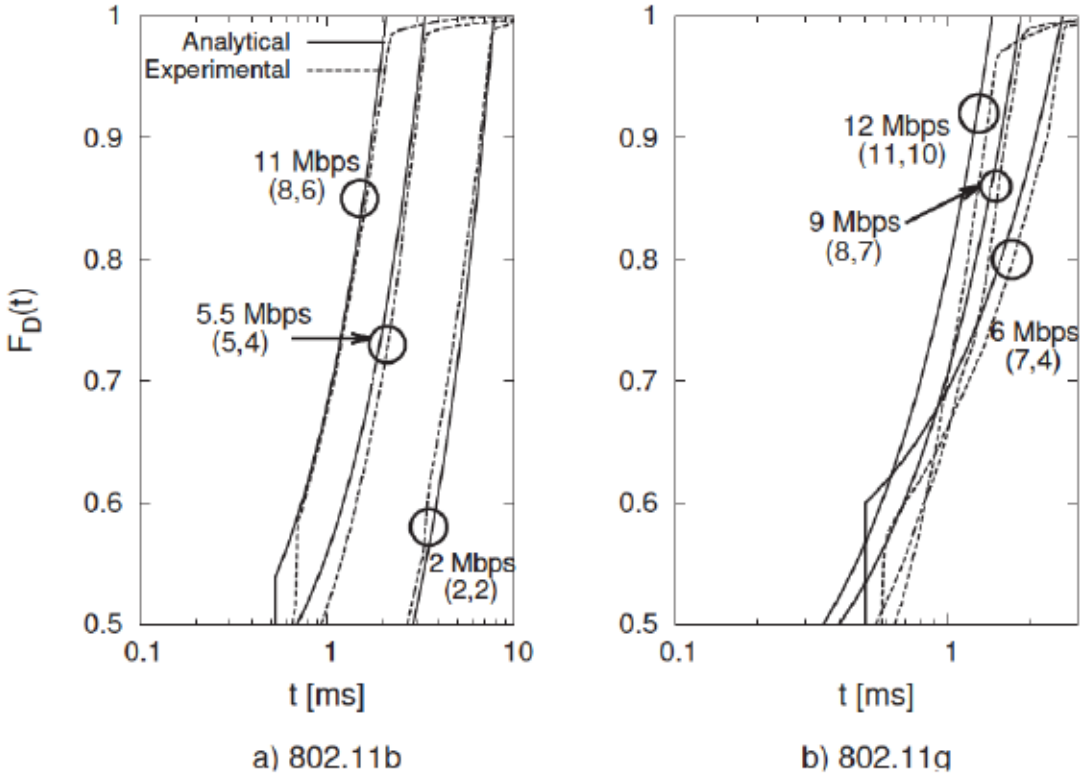


Figure 3.10: CDF of the voice delay for voice and data scenarios.

below this value. Note that, in all considered scenarios, the experiments are run with the maximum number of voice conversations supported by the WLAN, which are the most stringent conditions, providing worst case results for the delay.

Voice and data scenarios. We next consider the case of the WLAN with voice and data stations. Like in the previous case, we compute the CDF of the total delay of voice traffic in the downlink direction. Results are given in Figure 3.10. For each MCS considered in the figure, we select one specific point in the capacity region $\mathcal{C}$, which is the pair (n_v, n_d) reported in the figure below the MCS (e.g., for the case of 11 Mb/s, the scenario is $n_v = 6$, $n_d = 8$).

Our results show that, as expected, the activity of data stations worsens the performance of voice traffic, even when the number of conversations is smaller than n_v^*. The results also confirm that channel deferral because of data transmissions is significantly impacted, while the impact of collisions (responsible for the rarely occurring long delays, which appear in the CDF's tails) is practically negligible. In any case, and despite increased activity in the channel, the CDFs show that voice traffic still receives good service, since delay figures are in the range of a few ms. There results thus confirm that VoIPiggy is able to make room for more voice conversations in the WLAN while maintaining a good service level.

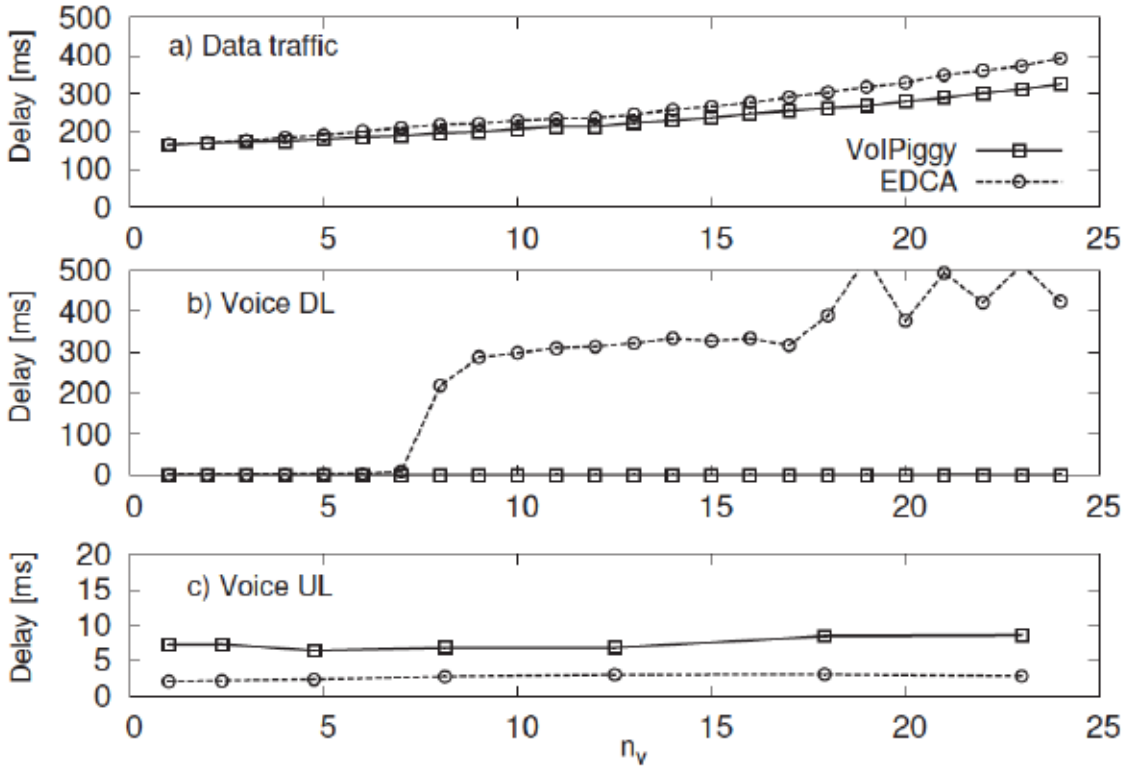


Figure 3.11: Delay for voice and data traffic.

In order to gain further insight regarding the delay performance of a WLAN with voice and data stations, we consider a scenario with n_v voice conversations and one station sending data to the AP in saturation, and then measure the average delay of the data and voice traffic (in the uplink and downlink). The results, given in Figure 3.11, show that not only does VoIPiggy improve the delay performance of voice traffic, but it also improves the delay performance of data traffic. Therefore, even though VoIPiggy gives higher priority to voice traffic as a result of making a more efficient use of the WLAN resources, it also improves the performance of data traffic in terms of delay (as shown in this section) and throughput (as shown in the next one).

3.5.4 TCP traffic

For the case of the data stations, in the previous scenarios we have assumed UDP uplink traffic to the AP. In this section we analyze the performance of VoIPiggy when TCP is used instead of UDP. To this aim, we set up a WLAN scenario in which n_v voice conversations are present and one data station is receiving TCP traffic from the AP.[11] For each scenario, we run the experiment for 60 s and compute the average TCP

[11]We also performed experiments with the data station sending TCP traffic, obtaining very similar results to the ones reported here.

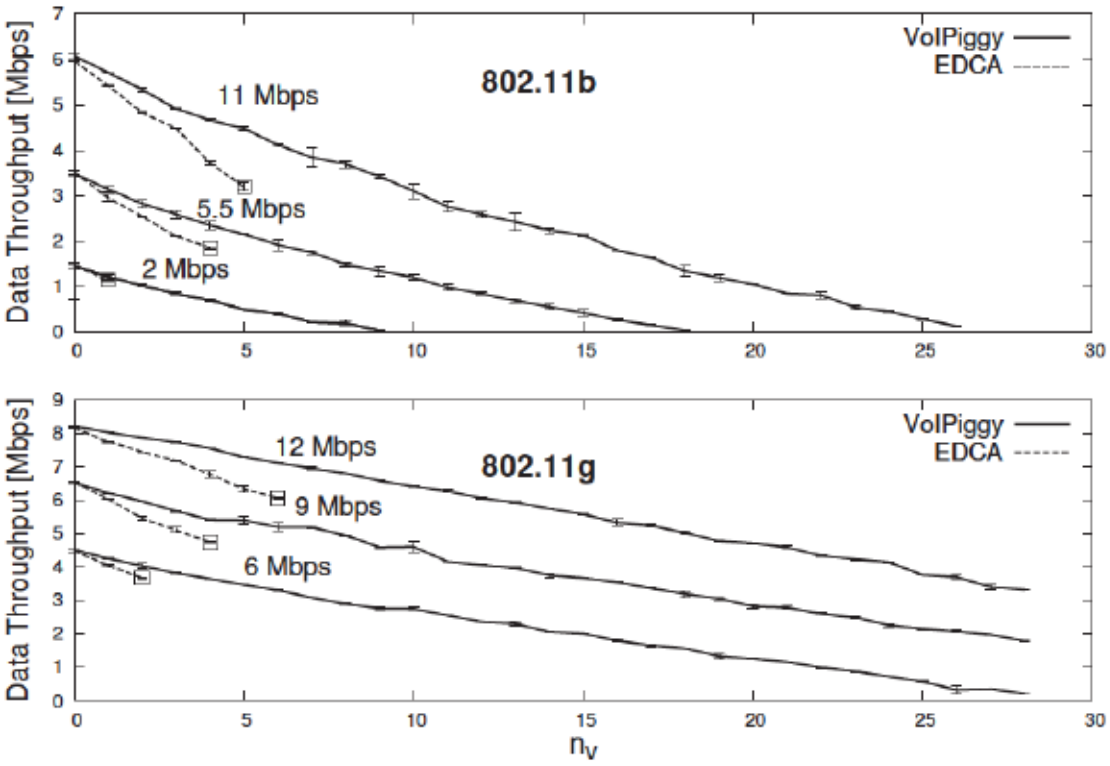


Figure 3.12: TCP throughput with n_v voice conversations (the figure only shows those points for which voice traffic does not suffer losses above 1%).

throughput obtained, repeating the experiment 5 times. We present the obtained results in Figure 3.12, with VoIPiggy (solid line) and EDCA (dashed line), for an increasing number of conversations, until n_v reaches the maximum number of stations (29, which is reached with VoIPiggy in the 802.11g case) or voice losses are above 1% (this only happens with EDCA).

There are several conclusions that can be drawn from the figure. First, the use of VoIPiggy effectively protects the performance of voice traffic, as the maximum number of voice conversations that can be supported reaches the values obtained in voice-only scenarios (n_v^*). In contrast, EDCA only supports a much smaller number of conversations than in the previous case, due to the increased aggressiveness of the data traffic. Second, TCP traffic obtains a larger throughput with VoIPiggy than with EDCA for all values of n_v, which confirms the previous observation that VoIPiggy improves not only voice performance but also data performance. Third, since VoIPiggy gives higher priority to voice stations, if the number of voice stations reaches its maximum value, data stations starve. In the event we wanted to ensure a minimum throughput for data traffic, our analysis of Section 3.3 could be used to apply admission control and thus limit the number of voice stations in the WLAN.

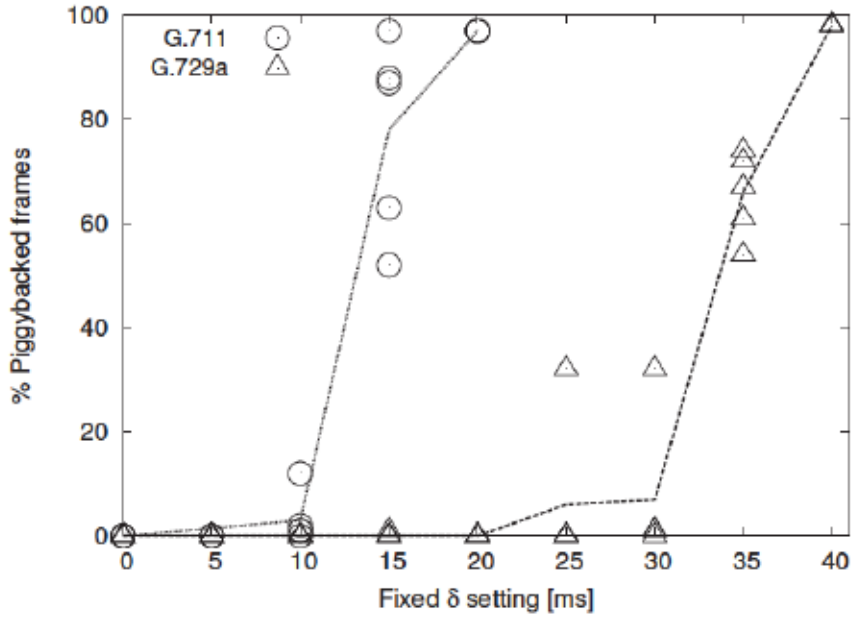


Figure 3.13: Percentage of piggybacked frames for a fixed configuration of δ (points represent individual samples of the experiments and lines correspond to their average values).

3.5.5 Validation of the algorithm to compute δ

Finally, we validate the ability of the algorithm presented in Section 3.2.2 to estimate the inter-arrival variability of voice frames which is required to hold waiting frames until a frame from the AP arrives while avoiding simultaneously unnecessary delays. To this aim, we use two Alix devices to set up a WLAN scenario consisting of one AP and one station, and compare the performance in terms of percentage of piggybacked frames for (*i*) a fixed δ setting, and (*ii*) the use of our adaptive algorithm.

We perform our evaluation using `mgen` to emulate the behavior of the G.711 and G.729a codecs, which generates 160 B frames every 20 ms and 40 B frames every 40 ms, respectively. We first analyze the case of a static setting of δ, for different values of this parameter. More specifically, we sweep from 0 to 40 ms, and for each δ value we compute the percentage of piggybacked frames during a 60 s experiment, running 5 repetitions for each one. The results are depicted in Figure 3.13 using one point per repetition (for ease of visualization we use lines to represent the average of the experiments). They confirm that, depending on the codec used, a different value of δ is required to maximize the number of piggybacked frames (i.e., $\delta \geq 20$ ms and $\delta \geq 40$ ms for G.711 and G.729a, respectively), and therefore a fixed setting of δ would require manual tuning.

We next analyze the performance of the algorithm to dynamically adapt δ, following a similar procedure as in the previous case (i.e., 5 experiments of 60 s each). The results, summarized in Table 3.3, show the average percentage of piggybacked frames of the 5

Table 3.3: Performance of the algorithm to compute

Voice codec	% piggybacked avg.		T_i (ms)	v_i (ms)
G.711	99.22	0.90 %	19.8	0.58
G.729a	99.12	0.73 %	39.4	1.10
Skype	99.61	0.17 %	30.3	5.76

experiments and its standard deviation ( ), along with the average values of the T_i and v_i parameters of our algorithm. The numerical gures con rm the ability of the algorithm to adapt to the variability of each frame arrival process, as practically all frames are piggybacked without using overly large values for .

To con rm that our algorithm also works with real voice applications, we evaluated its performance with Skype. The results, given in Table 3.3, show that practically all frames are piggybacked in this case, which con rms that VoIPiggy works with Skype. This con rms that Skype does not use silence suppression, as otherwise many of the uplink frames would not be piggybacked. In addition, we con rmed experimentally that Google Hangouts does not implement silence suppression either.

To evaluate the reliability of VoIPiggy, we have run several experiments and measured the number of frames that had to be retransmitted as well as the total number of lost frames. For all experiments, out o (approx.) 1500 frames, about 50 had to be retransmitted (less than 5%), and none was lost. We conclude that VoIPiggy is able to reliably deliver frames in both directions.

3.6 Summary

In this chapter, we have designed, implemented and evaluated VoIPiggy, a mechanism to improve the e ciency of MAC operation when voice tra c is present in 802.11 WLANs. In contrast to legacy operation, which incurs a very large overhead and wastes substantial time in contention, VoIPiggy extends the control frames sent from the stations to the AP with user data, thus practically halving the time required to transmit voice frames. In this way, not only the capacity to support voice tra c in the WLAN is boosted, but also data tra c bene ts from the increased e ciency in the delivery of VoIP.

We have described the small set of modi cations required to implement VoIPiggy, which are supported by existing COTS devices, and provided an analytical model to predict its performance in terms of capacity region and delay. To assess the performance improvements of VoIPiggy and validate its modeling, we have deployed a large-scale testbed consisting of 30 devices. Through extensive performance evaluation we have shown that our mechanism dramatically improves performance which provides a strong empirical support for the adoption of VoIPiggy and have demonstrated the accuracy of our

analytical model.

In the next chapter, we tackle the challenge of providing e cient video multicast delivery in the 802.11 networks. To that purpose we detail the operation of the GATS mechanisms of the recent 802.11aa standard and we evaluate them by means of simulations.

Chapter 4

Multicast Video Streaming in 802.11 networks: GATS mechanisms

In this chapter we detail the operation of the GATS mechanisms de ned in 802.11aa, namely DMS, Unsolicited Retries (UR) and BA. We pay attention to the di erent con g-urable parameters proposed by the GATS mechanisms, which correspond to the number of frames in a burst for the case of BA, as well as the number of times a frame is re-transmitted in the case of UR. Finally, we evaluate by means of simulations the GATS mechanisms in terms of two metrics, reliability and cost per service, which will be intro-duced in Section 4.3.

4.1 Motivation

As described in Chapter 1, IEEE 802.11 WLANs did not support e cient delivery of video streams. Subsequent amendments have triggered their widely deployment, due to its reduced price and high bandwidth, becoming the technology of choice to connect di erent devices in, e.g., residential deployments. Although the new IEEE 802.11ac, 802.11ad amendments provide a large increase in terms of achievable bandwidth, the new demands due to the arrival of high de nition video impose a several burden on the current MAC scheme, in particular for the delivery of multicast tra c. In this chapter we initially present a summary on the main MAC mechanisms to support multimedia services over 802.11 WLANs. Next we present the implementation of the new IEEE 802.11aa amendment and thoroughly evaluate it under a wide set of scenarios.

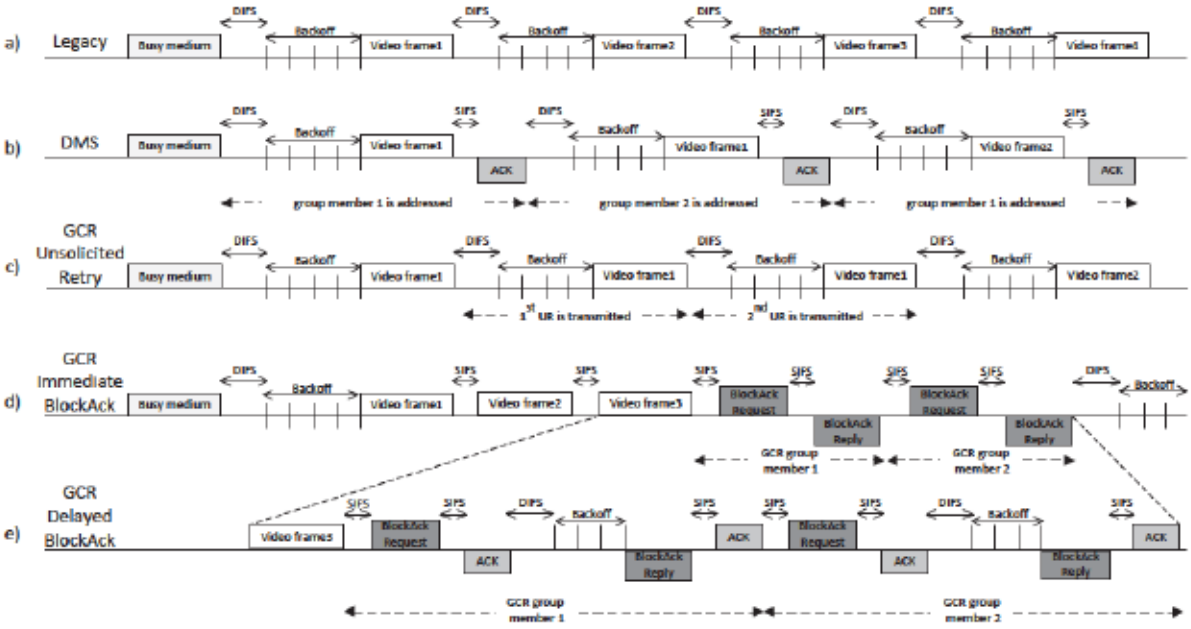


Figure 4.1: Mechanisms for multicast transmission in 802.11aa WLANs.

4.2 Description of the Group Addressed Transmission Service

Two of the main issues of multicast schemes defined by legacy 802.11, as described in Section 2.1.2, are: (*i*) the poor reliability of the service, since multicast frames are neither acknowledged nor retried, and (*ii*) the high inefficiency due to the use of a low modulation and coding scheme, as it uses the largest rate available in the Basic Rate Set. In order to overcome these limitations, the IEEE 802.11aa standard extends the set of mechanisms available for multicast transmissions. These mechanisms include the DMS, which was first introduced in 802.11v [56] and is extended to target group addressed frames, and the GCR service. The GCR service defines new retransmission schemes for *group* addressed frames, a group being a set of stations listening to the same (non-multicast) address, called *group concealment address*. With the specification of GATS, the number of mechanisms available to deliver traffic to multiple receivers in 802.11 WLANs is notably enlarged. To this aim, stations first have to form a "group", in order to agree on the address to listen to (the *groupcast concealment address*) and the specific mechanism to use. The group formation can be triggered by the AP or the clients, and they can leverage on newly defined frames (e.g., the *Group Membership Request Frame*) or the existing Internet Group Management Protocol (IGMP) protocol.

These mechanisms can be grouped into 2 schemes: *open-loop*, which are the ones that do not require traffic in the uplink direction; and *closed-loop*, which are the ones based on acknowledgments.

We next provide an overview of the mechanisms de ned by GATS, which we illustrate in Figure 4.1 for the case of two receivers, for the sake of comparison, along with the legacy 802.11 multicast service described in Section 2.1.2.1 (Figure 4.1a). We note that the legacy service never retransmits nor uses any type of feedback and x the MCS to maximum in the Basic Rate Set.

4.2.1 Direct Multicast Service (DMS)

This mechanism illustrated in Figure 4.1b, was speci ed by 802.11v to improve the delivery of management tra c, and 802.11aa extends it to support data frames. As the name implies, the scheme performs, for every group addressed frame and destination, a standard unicast transmission (as illustrated in Figure 4.1b for two groupcast members). The transmission follows the legacy procedure and the frames are retransmitted until an ACK is received, or the maximum retransmission limit is reached (and the frame is discarded). Although this mechanism provides very high reliability with a small implementation cost (as we will detail in Section 5.1.3), it also poses scalability issues as the consumed resources linearly grow with the number of group members: even with perfect channel conditions, the number of required transmissions per video frame is proportional to the number of receivers.

The DMS ensures reliability by retransmitting a frame as many times as needed, but it results very ine cient because each destination is addressed in unicast. In order to bene t from retransmissions but without so extreme ine ciency, 802.11aa speci es two new mechanisms that constitute the Groupcast with Retries (GCR) service: one mechanism that is relatively simple but not very e cient consisting of retransmitting the same frame R times, known as UR, and another mechanism that results more complex but improves the use of wireless resources levering the transmission of frame burst and BA polling scheme, namely BA.

4.2.2 GCR unsolicited retry (UR)

This delivery method, depicted in Figure 4.1c, *preemptively* retransmits all frames $R+1$ times, to lessen the impact of channel errors and increase the delivery probability to the STAs. In this way, the mechanism aims at improving reliability with a very simple scheme, which does not require a closed loop between the sender and the receiver(s), and therefore the price to pay is e ciency: successfully received frames can be retransmitted several times. Unlike the legacy service, this scheme may use also high MCS, hence it may be more e cient despite the unnecessary retransmissions. Note that the number of retransmissions (R) to use is not speci ed in the 802.11aa standard, but stated as implementation-dependent.

4.2.3 GCR Block Ack (BA)

The GCR Block Acknowledgment (BA) mechanism, depicted in Figure 4.1(d,e), extends the Block Ack operation of the standard to support multiple destinations. Its operation consists on sending a burst of up to GCR bu er size (we denote this bu er size as M) consecutive groupcast frames in a burst, and then performing a per-station polling operation to receive the corresponding acknowledgments. After receiving the *Block Ack Reply* frame from one STA, the AP sends another *Block Ack Request* frame to the next STA and so on, and then proceeds to send another burst of multicast frames.

GCR Block Ack comprises two variants depending on the control exchange procedure: GCR Immediate Block Ack and GCR Delayed Block Ack. In the *immediate* variant of the scheme (BA-I for short), depicted in Figure 4.1d, a backo process is executed for the transmission of the data burst, and the separation between frames is the minimum speci ed by the standard, namely, a SIFS time, i.e., the recipient of a Block Ack Request frame replies immediately (after a SIFS time) with a Block Ack Reply frame. In contrast, in the *delayed* version of BA (BA-D for short), illustrated in Figure 4.1e, each Block Ack frame is acknowledged by the intended destination, and all replies are performed after a backo operation. This backo procedure was introduced to enable stations with low CPU capabilities to use this mechanism, as it gives them more time to compute CRC values and generate the ACK bitmap. With the GCR Delayed Block Ack, both BlockAckRequest and BlockAck frames are acknowledged with an ACK frame. This delayed version imposes less stringent requirements on the implementation, as stations have at least one additional backo to process the received frame(s) and generate the corresponding ones, but results more ine cient and provides worse video service.

In contrast to the GCR UR scheme, the GCR BA introduces a notable implementation complexity, with a closed-loop between the transmitter and the receivers to account for acknowledged frames, and the use of a sliding window to keep track of the pending frames. This increased complexity, as we will see in Section 4.3, will lead to the a better e cient use of the wireless resources.

4.2.4 Parameters of the mechanisms

The mechanisms available with 802.11aa o er a variety of parameters to set, but the standard provides no guidelines towards their optimal con guration for any scenario (described in terms of, e.g., radio conditions, number of receivers, video BW). For simplicity, we set the MCS of data frames to MCS_D=54 Mb/s, and for the case of Control and Management frames we set it to MCS_C=24 Mb/s. We represent with $\mathcal{M}$ the set of mechanism under study, with $\mathcal{M}$ L UR_i DMS $BA\text{-}I_M$ $BA\text{-}D_M$, which therefore consists on the following ve schemes:

L: Legacy multicast, transmitted at MCS_C[1].

UR_R: UR with R retries (R 0) at MCS_D.

DMS: at MCS_D.

BA-I_M: with a maximum burst of M frames, at MCS_D.

BA-D_M: with a maximum burst of M frames, at MCS_D.

4.3 Performance evaluation

To perform the evaluation of the new multicast mechanisms, we use the OMNeT++ simulation framework,[2] con gured with the 802.11e EDCA module, which we extend in order to support the new schemes. For all results we present the average of 10 simulation runs, and we con rmed that 95% con dence intervals were below 1% of the average.

4.3.1 Scenario and channel model

Our WLAN scenario consists on one AP sending a video ow towards K destinations, using the 802.11g PHY layer. Our video ow is based on a trace- le from a movie clip of 2-minute duration encoded with the H.264 standard [57], one of the most popular video codecs [58]. Its spatial resolution is xed to 720x480, using a variable q parameter with constant bit rate of 3.91 Mb/s. We refer to the direction from the AP (stations) to the stations (AP) as downlink (uplink). Stations are 40 m away from the AP and con gured with a transmission power of 20 dBm. Our channel model closely follows the OFDM model described in [59].

In order to properly analyze the robustness and cost of the use of the 802.11aa mechanisms under a wide variety of scenarios, in addition to the above *channel losses* we also introduce, following [60], *interference losses*, to account for the impact of, e.g., hidden nodes. Based on this, in our simulations we will provide results based on the per-frame loss probability p, which accounts for both radio losses (with probability p_R) and interference losses (with probability p_I), as follows:

$$p = 1 \quad (1 \quad p_R)(1 \quad p_I)$$

4.3.2 Open-loop schemes

Here we analyze the performance of those mechanisms that do not require tra c in the uplink direction, which we refer to as *open-loop* schemes (i.e., $\mathcal{M}$ L UR_i). We rst analyze their *reliability*, i.e., how good they are at successfully delivering frames.

[1] This is the maximum MCS available in the Basic Rate Set.

[2] `http://www.omnetpp.org`

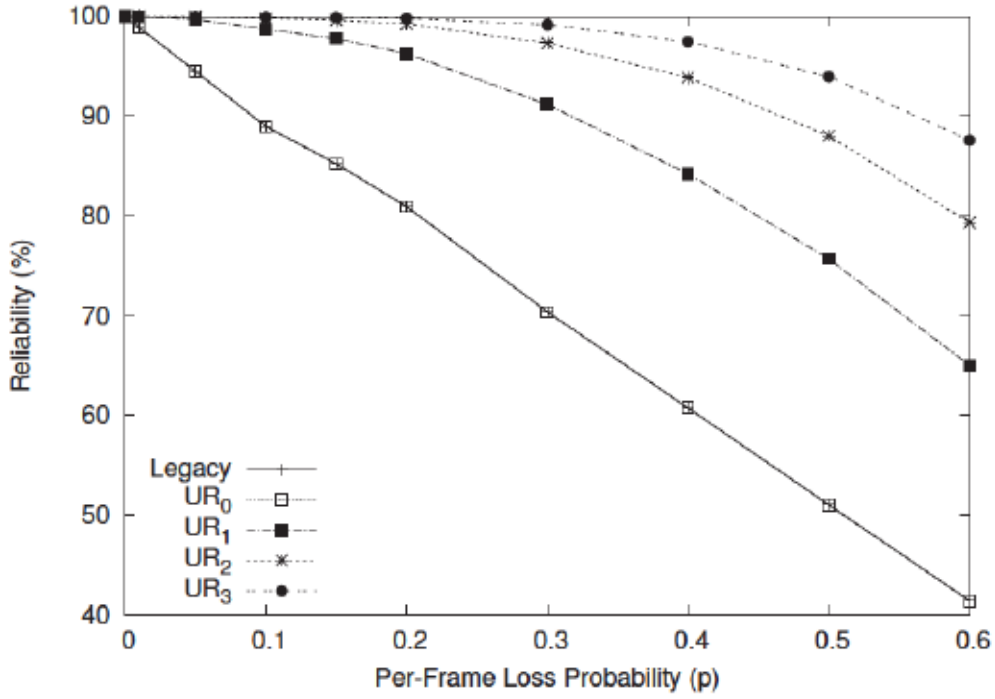


Figure 4.2: Reliability of the open-loop schemes.

More specifically, with K as the set of stations, if N frames were sent in the WLAN and out of them the subset N_i were successfully received by station i, the reliability of the scheme $\rho_{\mathcal{M}}$ can be computed as:

$$\rho = \frac{\frac{1}{K}\sum_{i=1}^{K} N_i}{N}, \tag{4.1}$$

which can be interpreted as the probability that a given video frame is successfully delivered to all stations. We plot the reliability of the considered open-loop schemes for a 5-receiver scenario (i.e., $K = 5$) and different values of p in Figure 4.2.

We use solid lines to represent the legacy mechanism, and dotted lines for the GCR unsolicited retry mechanism.[3] The results confirm that the legacy mechanism has the worst reliability, which, as expected, goes with $1 - p$ (e.g., a 60% reliability for $p = 0.4$). In contrast, the UR_i mechanisms improve reliability as i increases. Indeed, for the same value of $p = 0.4$, with UR_3 reliability is practically 100%. Even for the case of only one retransmission (UR_1), the reliability is improved by approximately 25% on average.

It should be noted that the above discussion does not take into account the *price to pay* to obtain a given reliability figure. In order to perform a *fair* comparison between the schemes, we need to take into account their corresponding *cost* in terms of resource consumption, as it is evident that, e.g., for good channel conditions, the legacy scheme

[3]Note that for every p value the case UR_0 matches the legacy mechanism in terms of reliability.

Table 4.1: Overhead of the open-loop schemes

	UR_0	UR_1	UR_2	UR_3	L
	1.00	2.00	3.00	4.00	2.25

will incur into a huge resource wastage due to the overly robustness of the MCS, or with UR_3 most of retransmissions will be unnecessary.

To quantify resource wastage, we de ne the *overhead* as the ratio between the channel time required by the mechanism $\mathcal{M}$ to send the frames over the minimum channel time that would be required to send them (regardless of the achieved). Let $\mathcal{T}_{\mathcal{M}}$ denote the channel time occupied by $\mathcal{M}$ to deliver the frames. Given that the GCR unsolicited retry with $R = 0$ (UR_0) is the scheme that consumes the least amount of resources, we can compute the overhead as:

$$= \frac{\mathcal{T}_{\mathcal{M}}}{\mathcal{T}_{UR_0}} \tag{4.2}$$

where we compute $\mathcal{T}$ by adding the time that the medium is busy (re)transmitting data and control frames.[4]

We report in Table 4.1 the overhead values of the open-loop schemes, which do not depend on the number of receivers (in contrast to the mechanisms of the next section) and, as expected, grow linearly with the number of retries. Given the MCS considered (54 Mb/s and 24 Mb/s for data and legacy, respectively), in this case we have that with two retries (i.e., UR_3) the scheme already consumes more wireless resources than the legacy case, despite its low MCS. We con rm that the overhead of the GCR unsolicited retry for an $R = 1\ 2\ 3$, are two, three and four times more than case without retries (UR_0).

With the above, we have on the one hand to quantify how good a mechanism is at delivering data, and on the other hand, to quantify how much it costs to use it. In order to better compare the schemes, we introduce the ratio as the *cost per service* of a scheme, i.e., the amount of resources consumed per each unit of reliability obtained. The results are depicted in Figure 4.3.

The results show that the legacy scheme (solid line) provides a cost per service that notably increases as p increases. The reason is that although the overhead is xed, the reliability decreases with p, and therefore grows. The same qualitative behaviour is obtained for the UR_i schemes, although in these cases the decrease of is less steep, therefore the cost per service is atter. The gure also serves to illustrate that depending on the number of retries con gured, the cost per service could be higher or smaller than the one with the legacy scheme, which illustrates the exibility introduced by 802.11aa and motivates the design of mechanisms to adequately tune this parameter. Finally, it

[4]Note that, the overhead cost is not the only cost that can be associated with a given scheme, as there are other costs, e.g., complexity/CPU, memory. We postpone the discussion on those until Section 4.3.5.

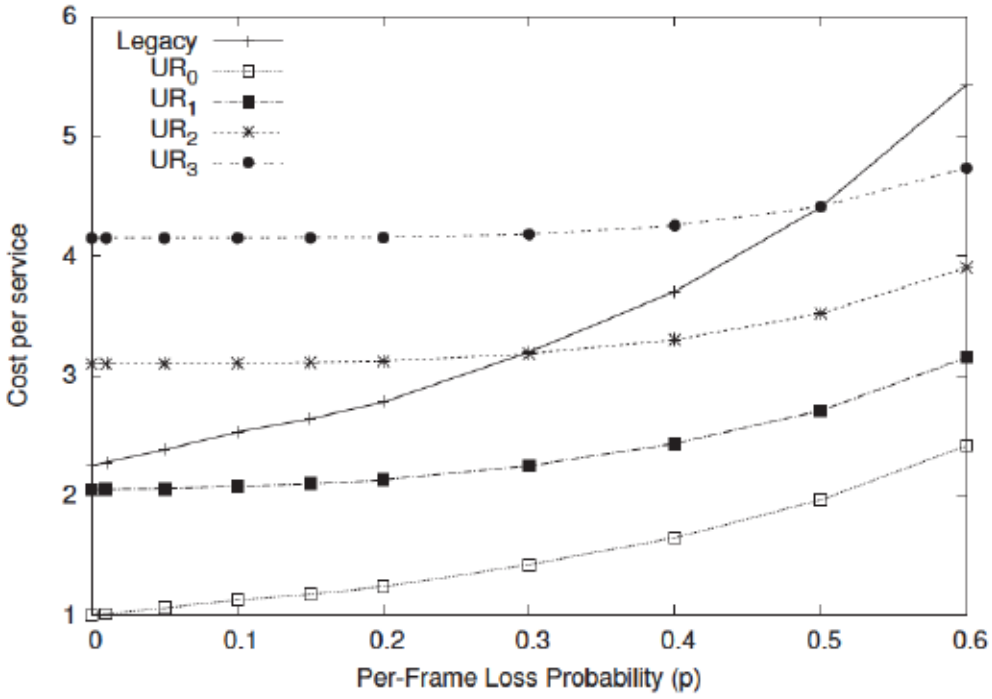


Figure 4.3: Cost per service (η/ρ) for the open-loop schemes.

should be noted that this (η/ρ) variable provides an adequate metric on the efficiency of a given scheme, but it does not take into account the actual performance experienced by a video flow. In Section 4.3.5 we will take this into account when comparing open and closed-loop schemes.

4.3.3 Closed-loop schemes

In this section we analyze the performance of those mechanisms based on acknowledgments, i.e., $\mathcal{M} \in \{\text{DMS}, \text{BA-I}_{\mathcal{M}}, \text{BA-D}_{M}\}$, whose performance therefore depends on the number of receivers K. We start our evaluation with the computation of the reliability ρ for the same WLAN conditions as in the previous section ($K = 5$). It should be noted that, with this schemes, and in particular with DMS, the need to retransmit a frame until it is acknowledged can severely choke the available bandwidth, up to a point where losses are also caused by frame drops at the output queue (we limit this queue to 200 frames).

We depict in Figure 4.4 the results for the reliability ρ of the closed loop schemes. We use solid lines to represent the DMS mechanism, and dotted lines for all GCR Block Ack mechanisms, whose performance in terms of ρ is very similar for the considered cases (note that for both schemes we use $M = \{5, 20\}$, to understand the impact of the maximum frame burst). Compared to the open-loop results in Figure 4.2, in this case ρ figures are much higher in general, thanks to the feedback from the stations. For the case of DMS, reliability is well above 95% until the frame loss probability reaches 0.3 –from this point

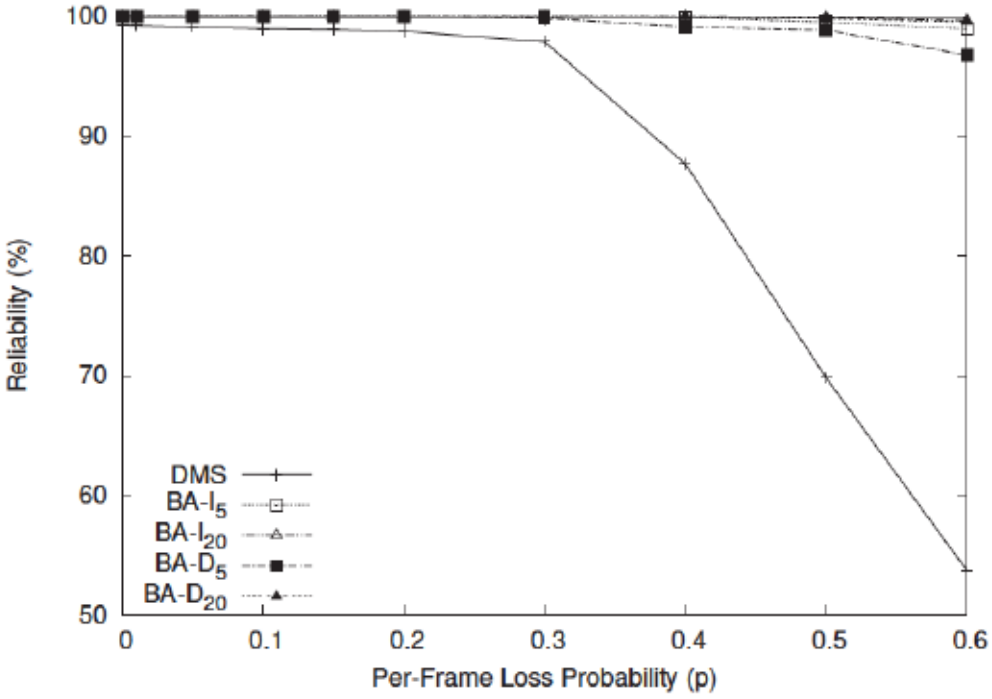


Figure 4.4: Reliability of the closed-loop schemes.

on, drops at the output queue dominate the loss process. For the case of Block ACK schemes, reliability is practically 100% for all p values below 0.5, and it decreases for the case of the delayed scheme with $M = 5$ to approximately 97% when $p = 0.6$.

We next compute the overhead η of each mechanism for the same scenario, with the results illustrated in Figure 4.5. As expected, the DMS mechanism presents a significant overhead, which is at least five times larger than the UR_0 (note that in this scenario $K = 5$). The GCR Block Ack mechanisms, in contrast, presents a significantly smaller overhead and very similar for all configurations (a closer look to the figure reveals that $BA - I_{20}$ shows the smallest overhead).

We next compute the cost per service η/ρ for the same scenario ($K = 5$) and present the results in Figure 4.6a. The figure presents the same qualitative results as the ones obtained for η, which is caused by the relatively constant η values.[5] Furthermore, it also confirms that for $K = 5$, there are little differences between the considered Block Ack schemes.

In order to understand the differences between Block Ack schemes, we provide the (η/ρ) values for $K = 10$ and $K = 20$ receivers in Figure 4.6b and Figure 4.6c, respectively (note that we only provide the DMS values for $K = 10$, due to its scalability issues). The figures show that, as the number of receivers increases, differences arises between

[5]It is worth noting that with open-loop schemes we have a fixed overhead, while in this case we have a relatively fixed reliability.

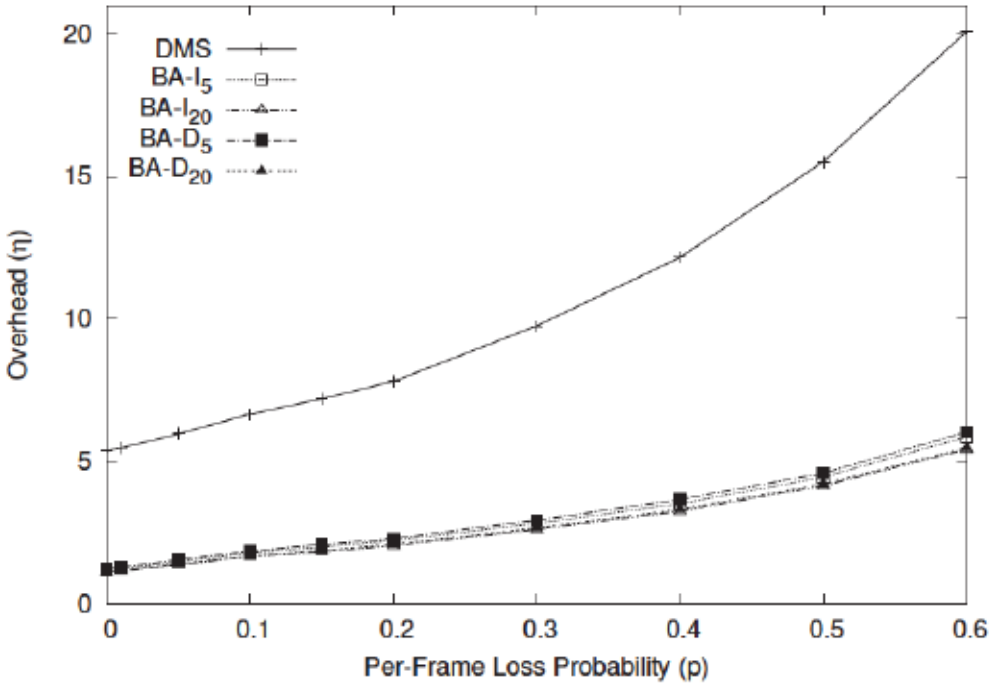
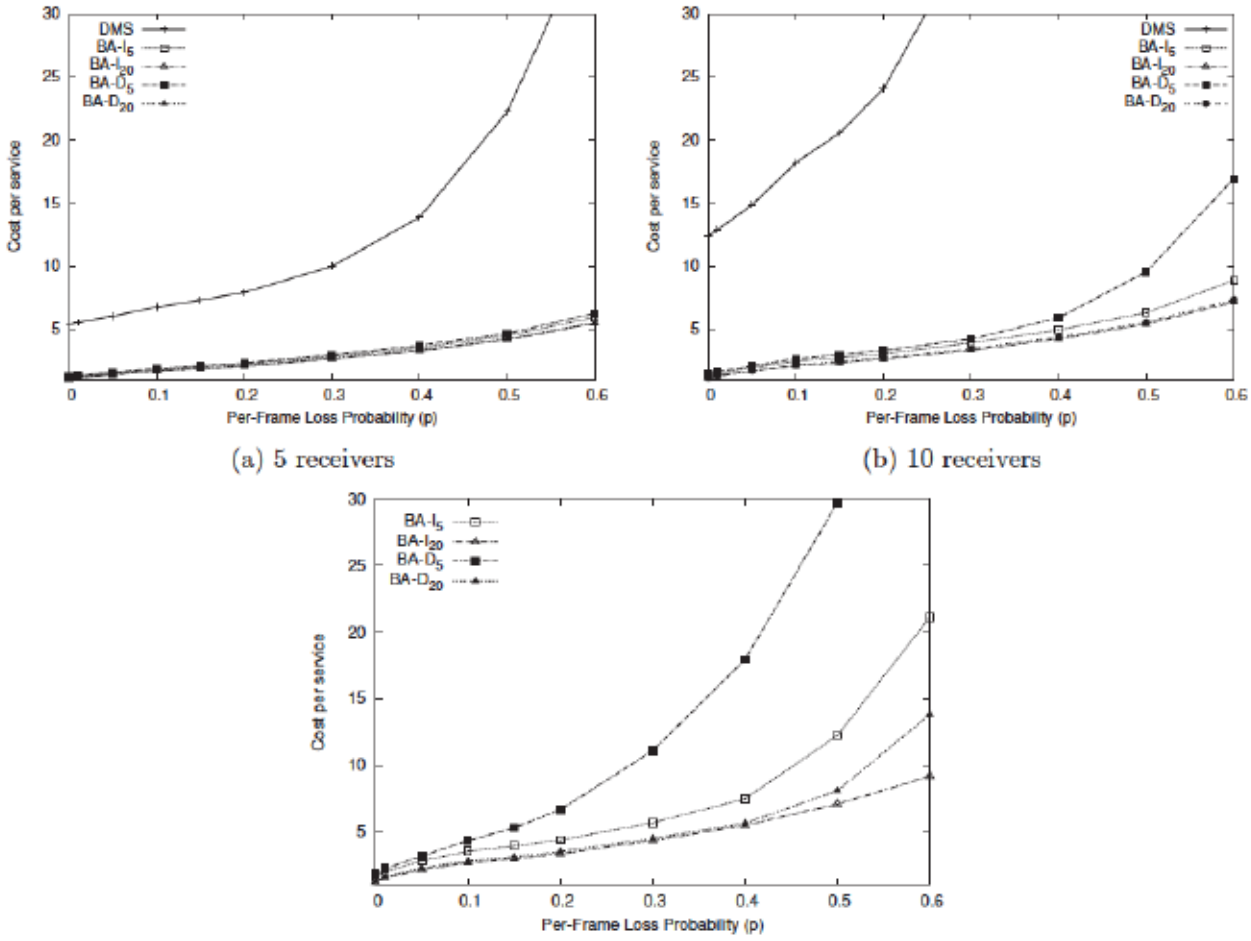


Figure 4.5: Efficiency of the closed-loop schemes.

BA schemes: the delayed version with a maximum burst of $M = 5$ frames provides the largest cost per service, while the immediate version with $M = 20$ provides the smallest. Furthermore, from the results and configurations considered, the choice of the maximum burst length M has a more significant impact that the difference between the immediate and the delayed versions of the Block Ack schemes. We can point that the lower the burst size, the larger the airtime overhead. As the reliability is around 100%, regardless of the burst size, the Delayed Block Ack with a burst size of 5 frames turns into the least convenient.

4.3.4 Comparing open and closed-loop mechanisms

Here we evaluate the efficiency of all considered schemes when a minimum QoS is guaranteed. More specifically, we set a threshold on the reliability ρ, and compare the cost per service of each mechanism as long as their performance is above this threshold. As it is well known, the impact of the packet losses depend on the loss pattern that the application is experiencing, i.e., burst or isolated. In addition, it is directly dependent on the type of lost frames (B, P or I frames) and the codec applied [61]. According to [62], typical Internet packet loss rates are 3%, 5%, 10%, and 20% , as well as in [63] packet loss rates larger than 30% mean a high degradation in the video. Following [64,65], which report that video quality is severely degraded when loss rates are above 5-10%, we fix 10% as the minimum requirement on reliability, hence we provide a rather conservative



(a) 5 receivers

(b) 10 receivers

(c) 20 receivers

Figure 4.6: Cost per service (η/ρ) for the closed-loop schemes for K=5, 10 and 20.

validation of the mechanisms studied. Note that the selection of this quality threshold will directly depend on several parameters, such as: type of codec, the video requirements and the WLAN scenario. Based on this, we plot in Figure 4.7 the cost per service of the considered schemes. For the sake of clarity we do not depict in the figure the case of DMS, whose cost is significantly higher than any other scheme. In addition, we do not either depict the case of Delayed Block Ack, given that its performance is very similar to the one obtained with the Immediate version.[6]

We can split the analysis of the results in Figure 4.7 in two parts, namely, when losses are very small ($p \leq 0.05$) and when loses are significant ($p > 0.05$). For the first case, UR_0 achieves the minimum cost per service, and in contrast to the Block Ack schemes, its performance is independent of the number of receivers K. Indeed, BA schemes are *second best* when $K = 5$, but for $p \leq 0.05$ it is interesting to observe that UR_1 achieves a

[6]As explained before, the choice between the Delayed or the Immediate version will mostly depend on hardware constraints.

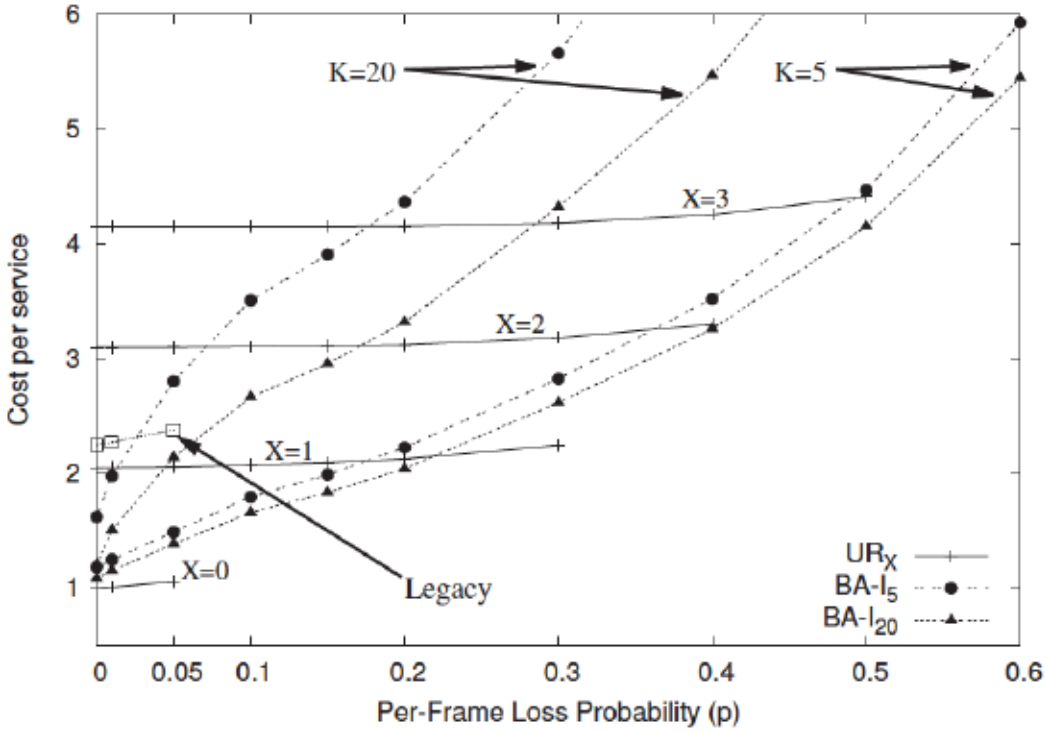


Figure 4.7: Cost per service (η/ρ) for multicast schemes for $\rho \geq 90\%$.

smaller cost per service than BA when the maximum burst size $M = 5$ and $K = 20$, and that even the legacy scheme is more efficient than BA in that case.

As losses become significant, the UR scheme has to increase the number of retries to provide the required reliability, up to a certain point where it cannot achieve $\rho > 90\%$. For instance, UR_1 provides the best performance for $p = 0.3$, but from this point on, it cannot guarantee enough reliability. Similarly, for $p = 0.4$ and $p = 0.5$ the cost per service of UR_2 and UR_3 is very similar the minimum one, which confirms its good efficiency in terms of cost per service, but beyond these probabilities the video flow suffers from severe losses. These results confirm that a key issue when using UR is to properly tune the number of retransmissions, as depending on WLAN conditions this could result in a very inefficient scheme (e.g., UR_3 when $p < 0.1$) or one of the most efficient ones (e.g., UR_2 for $p \approx 0.1$), which in addition has a performance that does not depend on the number of receivers.

Considering the Block Ack schemes, the figure shows that for the case of a small number of receivers ($K = 5$) their cost per service is relatively small, being able to provide enough reliability for all the considered values of p. When the number of receivers increases, the overhead associated with the polling scheme becomes more evident, leading to the largest values of (η/ρ) for $p \geq 0.2$.[7] Furthermore, in these circumstances (i.e., when

[7]Note that the DMS mechanism is not considered due to its high costs.

the control tra c is signi cant) the maximum burst length M has a notable impact on performance, as opposed to the case of $K = 5$ receivers, where there are little di erences between the two M con gurations.

The above results quantify the di erences in terms of performance between the multicast schemes. We next summarize qualitatively their di erences, discussing the pro s and con s of each scheme.

4.3.5 Discussion

Here we summarize the main results from our simulation-based assessment while taking into account the qualitative di erences between the mechanisms, both in terms of their ease of use (in case, e.g., they require a proper tuning of a parameter to achieve good performance) and the complexity introduced.

The *legacy* mechanism, as expected, is the approach that provides in general the poorest reliability, and given its low MCS, it also provides very low e ciency values. However, it should be noted that in terms of cost per service, it can outperform very aggressive con gurations of UR_i, and that for good WLAN conditions[8] it can outperform Block Ack schemes when the number of receivers is high. Given its non-existent implementation complexity, we believe that still it cannot be discarded as a valuable scheme to transmit multicast tra c, although its use is limited to very speci c conditions: large number of receivers, some with weak radio links as they require a low MCS, and low tra c activity in the WLAN.

The *GCR unsolicited retry* mechanism o ers a relatively large variety in its performance, which obviously depends on the number of retries con gured but not in the number of receivers. The latter is a major advantage of the scheme, in contrast to closed-loop mechanisms, and we have seen that even for moderate frame loss probabilities its performance can be as good as the one obtained with BA (for 5 receivers) or signi cantly better (for 20 receivers). Although the mechanism does not introduce signi cant overhead in terms of implementation, its main weakness is that it requires a proper tuning of the number of retries, as otherwise the above good features are di cult to achieve.

The *Directed Multicast Service* mechanism poses severe scalability issues, as its overhead is at least the number of receivers, and greatly increases with frame losses due to retransmissions, causing frame drops at the output queue. Its implementation complexity is, on the other hand, moderate, the unicast mechanism is already available with existing interfaces and the only modi cation is basically to replicate each frame towards each intended destination. Based on this, the DMS is only suited

[8] We remark here that p accounts for both radio conditions and collision from other tra c sources.

Table 4.2: Overview of the schemes available with 802.11aa

Scheme	Complexity	Reliability	Overhead	Cost per service	Scalability	Target scenario
Legacy	Low	Low	Medium	Medium	High	Legacy stations, good WLAN conditions
UR_R	Medium	Medium[a]	Low[a]	Low	High	Large number of receivers, relatively good WLAN conditions
DMS	Medium	High	High	High	Low	Good WLAN conditions, small number of receivers
BA-I_M	High	High	Medium[b]	Low	Medium	Moderate number of receivers, average WLAN conditions
BA-D_M	High	High	Medium[b]	Medium	Medium	Same as BA-I, interfaces with limited capabilities

for scenarios with very small number of receivers and high constraints in terms of reliability.

The *GCR Block Ack* mechanisms achieve the high reliability of DMS without its enormous overhead, at the expense of a high implementation complexity (both at the AP and the stations). Indeed, the AP has to keep a copy of all frames as long as they are not acknowledged, and the stations have to compute the CRC for each frame and send the feedback in the Block ACK, which requires not only a new control exchange but also relatively high computation capabilities if these are not available, the delayed version lessens this requirement. This complexity supports a good performance in terms of cost per service, although when the number of receivers is large, it can be outperformed by simpler mechanisms (properly con gured).

The main highlights of the above discussion are presented in Table 4.2, in which we compare for every scheme, its main features in terms of complexity, reliability, overhead and cost per service, and also we provide the target scenario. In this way, one of the main conclusions of our work is that there is no *clear winner* among the mechanisms available with 802.11aa, as each of them o ers a di erent trade-o in terms of performance and complexity, and therefore their use depends on the target scenario and its parameters.

4.4 Summary

In this chapter, we have performed an evaluation of the GATS mechanisms available with IEEE 802.11aa in terms of reliability, overhead and cost per service, and compared them to one another and against the legacy multicast service. We have con rmed that the new mechanisms signi cantly enrich the available choices to deliver video over WLANs, as they provide di erent trade-o s in terms of complexity, e ciency and reliability. We have identi ed the main limiting factors of each mechanism, which paves the way for the derivation of guidelines to optimally con gure the parameters of each mechanisms and to select the best suited for a given scenario.

Chapter 5

Implementation and Experimental Evaluation of 802.11aa GATS Mechanisms

In this chapter we implement the GATS mechanisms proposed in the 802.11aa standard, namely DMS, UR, BA. We provide the first available implementation using open-source firmware `OpenFWWF`, describing the modifications required. Finally we evaluate the performance of these new mechanisms under different scenarios and discuss the results.

5.1 Implementing GATS

A major advantage of GATS is that it does not substantially alter the functionality of the existing MAC, i.e., it uses standard features and timings of the legacy MAC. This ensures backwards compatibility with legacy stations and enables deployment of the new mechanisms on standard COTS hardware. Indeed, in this section we report how the new mechanisms can be implemented over this existing COTS hardware, using the existing open-source firmware `OpenFWWF` that has been used in the past to implement other extensions for 802.11 [54, 11, 12].

5.1.1 Platform used

We have implemented GATS on the same devices used in Chapter 3, Alix 2d2 boxes by *PC Engines*, which embed a Geode LX800 AMD 500 MHz CPU, 256 MB DDR DRAM, 2 mini-PCI slots and a Compact Flash socket. The software platform is Ubuntu 10.04 Linux (kernel 2.6.36) and as wireless chipsets Broadcom BCM94318MPG 802.11b/g, which supports the open-source driver *b43* and the `OpenFWWF` firmware.

The implementation consists, basically, on modifying three different modules: the `OpenFWWF` *firmware*, which is required for time-critical operations (such as, retransmis-

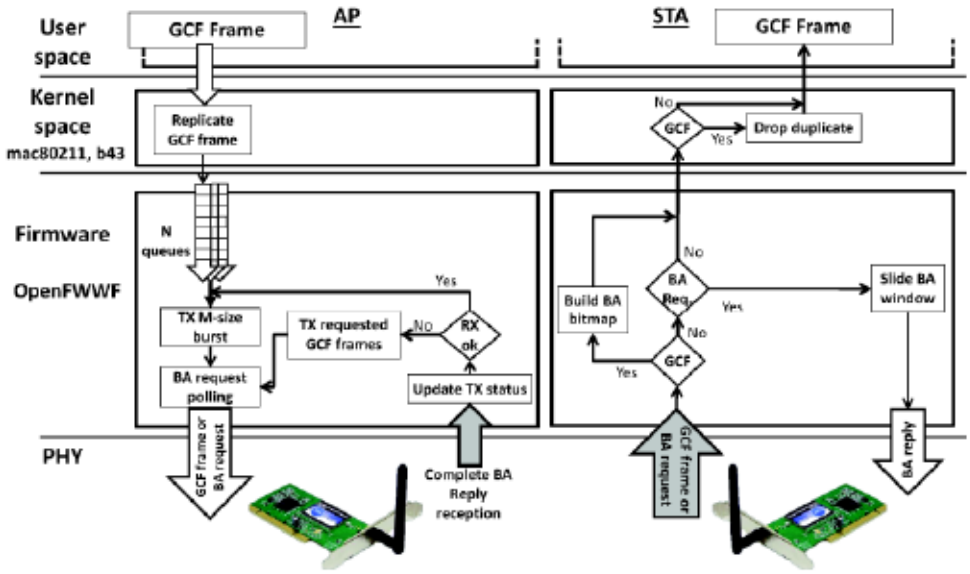


Figure 5.1: Implementation platform.

sions, ACKs), the *b43 driver*, for operations with less stringent requirements, and the *mac80211* module of the Linux kernel, so the modifications (e.g., duplicate packet detection) do not impact the behavior of other modules. We next describe the modifications required to implement each scheme. The software/hardware architecture in which we prototype GATS is illustrated in Figure 5.1.

5.1.2 Groupcast formation and management

We address in this section common configuration issues that affect equally to some of the implemented mechanisms. First, we statically configure the groupcast address as `BE:EF:BE:EF:BE:EF` for all the group members, undertaking the imposition that the less significant bit of the first octet must be 0, as this is not a multicast MAC address.

With respect to the group announcement and joining, we statically configure the groupcast detection and joining, with the presence of one group, but this is easily extendable to account for N different groupcast groups with different multimedia stream needs. The Block Ack service implies a negotiation of several parameters, e.g.: number of frames in a burst, starting sequence control and policy, which in our case are pre-configured. These values are defined at the firmware in the `BAR_DATA_LENGTH`, `SEQ_NUM_NEGOTIATED`, `BAR_POLICY` registers, respectively.

5.1.3 Required modifications

Our implementation of GATS requires introducing changes at the *firmware* and the *kernel* modules (i.e., driver and mac80211) of the device. We next provide a qualitative description of the required changes at these modules to implement GATS, and later

provide a quantitative evaluation of their complexity.

Directed Multicast Service This mechanism builds on top of the legacy service and therefore its implementation results simple. At the *transmitter* side, a copy of every frame from the application layer is generated for each intended destination, changing the corresponding MAC address. Given that these are not timely operations, the process is performed at the driver, and then the copies are passed to the transmission side. We keep a list of the associated stations by hacking the `b43_wl` struct at the driver level. At the *receiver* side, the unicast scheme guarantees that each frame is received only once, so no modi cations are required.

Unsolicited Retries This mechanism requires to transmit the same frame $R + 1$ times, with backo but without ACKs. Like before, these are not timely operations and therefore the main changes are performed at the driver level. At the *transmitter* side, ACK waiting is disabled at the driver by modifying the `b43_generate_txhdr()` function, similarly to multicast frames. The driver place the corresponding 802.11aa groupcast frame in a speci c queue, which is managed by the `select_ring_by_priority()` and `select_ring_by_priority_80211aa()` functions. The R retransmissions are programmed at the rmware, whose value is speci ed in the `NUM_RESEND_FRAME` register and being a con gurable parameter that we set up in real time (as detailed in Section 5.2).

At the *receiver* side the modi cations are three: (*i*), at the rmware for frames sent to the con gured groupcast address (which is not a multicast address) pass them to upper layers; (*ii*), replace this address with the stations one before passing the frame to the upper modules, which is done at the driver; (*iii*), handle potential duplicates at the mac80211 level (`ieee80211_rx_h_check()`).

Finally, for debugging and monitoring purposes, we extend the `b43_wldev` structure with transmission and reception statistics (e.g., total number of 802.11aa frames transmitted and received) to account for 802.11aa tra c.

Block Acknowledgment With this mechanism, the sender can transmit up to GCR bu er size consecutive frames in a burst (we denote this number by M), which can be either new frames or retransmissions. As the standard does not specify a policy to schedule these transmissions, we implement the following one. We wait for the queue to ll with M new frames, and do not admit new frames until all these frames are acknowledged by all receivers. In this way, if out of the M frames, N are positively acknowledged, the next transmission burst will consist on $M \quad N$ frames.[1]

The above requires at the *transmitter* the following changes. First, the driver collects M frames before copying the entire burst to all the hardware queues, so that the rmware

[1]This policy is a trade-o between always waiting for the output queue to be lled with M frames, which would maximize e ciency but could introduce overly large delays, and immediately sending frames as they are available, which would minimize delay but at the cost of large ine ciency. We note that, in order to prevent a potential HOL blocking caused by receivers with poor link qualities that require too many retransmissions, the AP should not use overly large values for the MSDU lifetime parameter.

Table 5.1: Implementation *cost* of each mechanism

Scheme	Node	New lines of code			
		Kernel	Firmware	Subtotal	Total
DMS	TX	149	0	149	149
	RX	0	0	0	
UR	TX	50	16	66	104
	RX	21	17	38	
BA	TX	577	618	1195	1632
	RX	132	341	473	

may draw the same burst multiple times for handling retransmission. The reduced size of the shared memory of the device, only 4 KB large, makes no possible to store all the frames in it. For that purpose, we use the four remaining FIFO queues provided by the driver, and corresponding to the QoS and EDCA support, in order to store the frames to retransmit. When a frame arrives to the driver is directly replicated to the di erent queues, guaranteeing that the synchronization is maintained among the di erent queues. This is done by modifying the `b43_op_tx()` function of the driver. We store the list of the groupcast associated stations by tweaking the `b43_wl` struct at the driver level. The rmware (re)transmits all frames of the burst that have to be (re)transmitted by spacing them of a SIFS, then it polls receivers with `Block Ack` requests in round-robin (in the order stations joined the groupcast). The Block Ack request frame is stored into the `SHM` and contains the sequence number of the rst frame in the burst. If the receiver does not send the corresponding reply, after a timeout expiration the AP will retransmit the Block Ack request up to seven times. Upon the reception of all replies, the AP updates its transmission status and check if any retransmission is required based on the bit-wise operation with the bitmaps collected from the groupcast members (ReTX map) and saved into the `SHM_BITMAP_80211_RETR_BUFF *` registers. The groupcast frames to be retransmitted are marked with a bit set to 1 in the ReTX map, while the others are simply discarded. When all the groupcast frames of a burst are correctly received, or until their lifetime expires, the rmware will ush the frames of the burst in the remaining queues. If a transmission phase is needed, only such frames are sent.

At the *receiver* side, the rmware updates the reception bitmap (i.e., correctly received frames in the burst) after receiving each frame, which leverages on the computation of CRCs and the sequence number. Each station stores the sequence number of the rst packet in the burst and a bitmap to keep track of the correctly received groupcast frames, being the bit *k-th* set to 1 if the frame is correct, or set to 0 otherwise. The corresponding registers that contain the previous information are: `RX_FRAME_ADDR3_2` and `SHM_BITMAP_80211AA_RX *` . This bitmap is sent in the `Block Ack Reply`, saved in the `SHM`, during the polling. Like in the UR case: groupcast frames received at the rmware are passed to the upper layer, duplicated frames (i.e., not received by other stations) are handled by the `mac80211` module, and the driver substitutes the groupcast address by

the stations .

Finally, for debugging and monitoring purposes, in the driver we extend the `b43_wldev` structure that contains various transmission and reception statistics (e.g., total number of 802.11aa frames transmitted and received, length of bursts, transmission attempts) to include those related to the operation of 802.11aa.

5.1.4 Implementation summary

We build each mechanism on the standard kernel MAC and driver modules, introducing no changes to the upper layers. The resulting implementation is fully compatible and compliant with the current Linux networking stack.[2]

Table 5.1 hints the implementation cost of each mechanism by reporting the number of lines of additional code required for implementing it. Although simply counting lines of code does not take into account other factors like the time needed to design and debug a prototype, still we believe that the table gives a fair quantitative evaluation of the complexity of each mechanism, as DMS and UR result similar in terms of complexity (although DMS requires changing only the transmitter) while BA is (at least) one order of magnitude more complex. As we will see in the next section through real-life experimentation, this increased complexity translates into higher reliability. However, this poses a trade-o as a simpler scheme can ful ll the requirements depending on the scenario.

5.2 Experimental results

5.2.1 Testbed description and set-up

We deploy a testbed, as illustrated in Figure 5.2 of 30 Alix 2d2 devices acting as wireless stations and one desktop machine acting as AP. The AP uses a 7 dBi omnidirectional antenna and the stations are equipped with 2 dBi omnidirectional antennae. All nodes use a transmission power of 10 dBm and the 802.11g PHY layer. We set-up a standard desktop machine as the source of video tra c when needed (not shown in the picture). Unless otherwise noted, we run all experiments in channel 14.

All nodes are equipped with a wired interface, which is used to set-up and control the experiments. Depending on the experiment, a set of N_v stations will act as multicast receivers and a set of N_d will act as data stations. For each test scenario, the AP and the N_v stations load through the wired interface our modi ed version of the 802.11 stack, and use the con guration of the backo parameters recommended by the EDCA standard for video. For simplicity, we statically con gure on these video stations the groupcast address to use (namely, `BE:EF:BE:EF:BE:EF`), as well as other control parameters such

[2]We refer the reader interested on the detailed description of the implementation to the technical report [66].

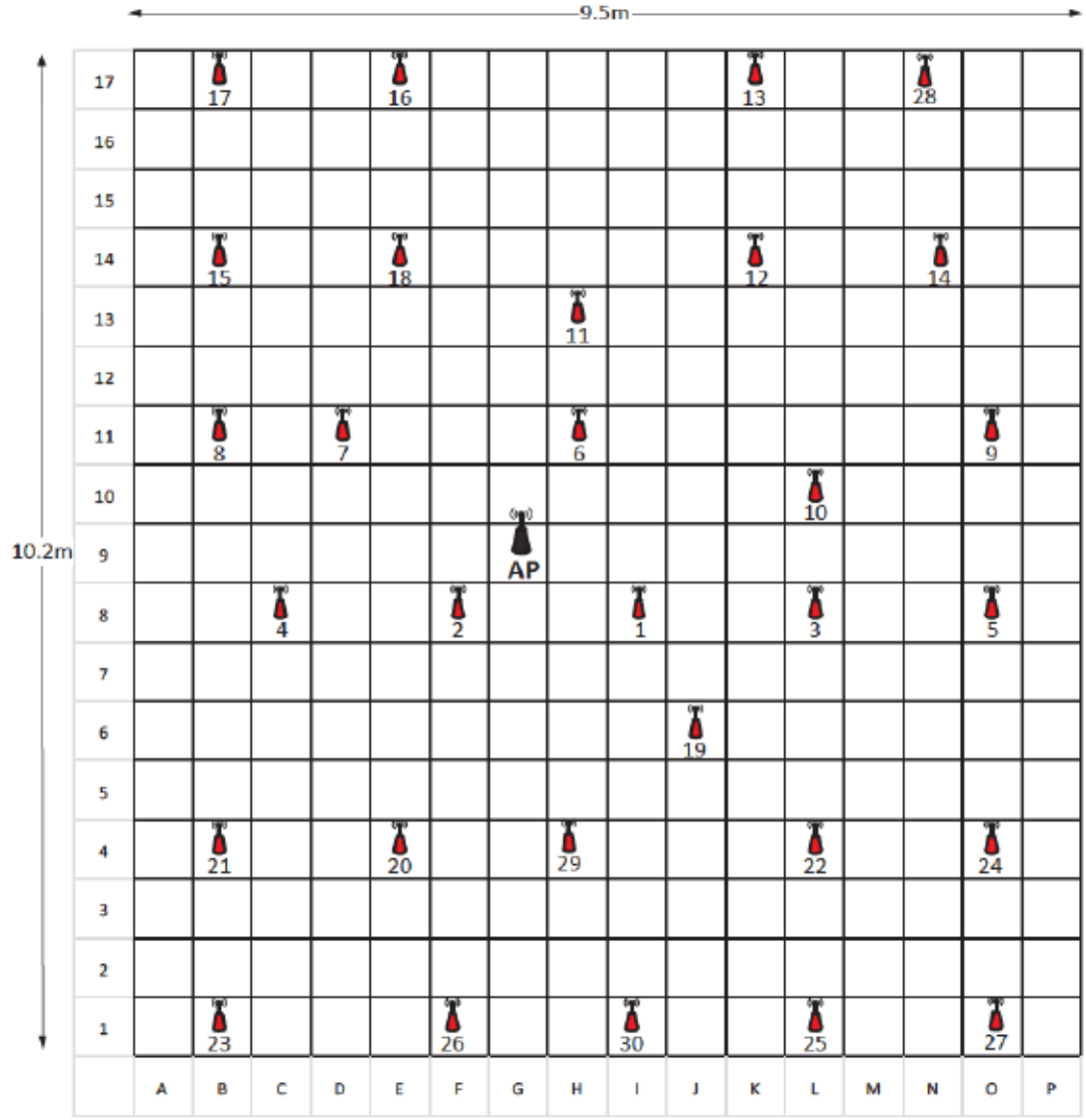


Figure 5.2: Testbed of 30 stations and 1 AP.

as the number of retries R for the case of UR, the maximum burst length M for BA, or the initial sequence numbers.

The N_d stations load the Broadcom 802.11 firmware and b43 driver, and use the legacy DCF MAC protocol. In this way, we assess the performance of GATS in a scenario with legacy stations, which do not support the service differentiation provided by EDCA. Our assessment therefore constitutes a "worst case" scenario were data stations are very aggressive –we left for future work the analysis and optimal configuration of 802.11aa scenarios were MAC parameters can be tuned. Similarly, data stations will generate uplink and saturated UDP traffic instead of TCP traffic, as the former results in very aggressive contention activity in the channel and, correspondingly, puts the most stringent conditions for the performance of GATS. Finally, throughout our experiments we fix the

MCS of data, groupcast and BA request and reply frames to 54 Mb/s, which represents the most stressing conditions for our implementation in terms of frame processing rates,[3] while for the legacy scheme is set to 24 Mb/s. This is also the MCS used for control tra c.

5.2.2 Synthetic tra c

We rst run some experiments with the `mgen` tra c generation tool, to assess the performance of the GATS mechanisms with di erent input loads. More speci cally, we con gure the AP for sending Constant Bit Rate (CBR) tra c of xed rate r towards $N_v = 10$ receivers, and $N_d = 10$ data stations constantly backlogged for sending UDP tra c towards the AP. Data payload of all frames is xed to $L = 1400$ B.

For each scenario we are interested in two performance gures, related to the *e ectiveness* and *e ciency* of the considered GATS mechanism, which are, respectively:

> Video delivery rate (VDR), which is the average throughput received by stations over the throughput generated by the sender. This gure quanti es the reliability of a given GATS scheme.
>
> Aggregated data throughput (ADT), which is the sum of the throughputs obtained by data stations, and serves as an indication of the amount of wireless resources left for data tra c (the higher the ADT, the more e cient the multicast mechanism).

We run 5 experiments of 30 s each for di erent values of r, and provide the average values of the VDR and ADT in Figures 5.3 and 5.4. For the case of VDR (Figure 5.3), results con rm that BA guarantees reliability for almost all r values (the higher the r the higher the M required to properly prioritize video over data), while DMS su ers from its poor scalability properties, failing to deliver even 50% of the tra c for $r = 3$ Mb/s. Between these two extreme cases, we see a variety of behavior vs. r: the legacy service outperforms UR with $R = 0$ due to the use of a more robust (and lower) MCS, but cannot deliver rates above 18 Mb/s because of the same reason. Similarly, con guring UR with two transmissions ($R = 1$) guarantees delivery for r below 12 Mb/s, but starting from this value su ers from frame drops at the transmission queue (for the case of $R = 0$, these drops start at 24 Mb/s). It is worth noting that, for these cases, $R = 1$ is *enough* to guarantee a good delivery rate, and setting overly large values of R (e.g., $R = 4$) not only provides no good, but even worsens performance due to the frame drops.

We analyze the ADT results, shown in Figure 5.4. As expected, using UR with $R = 0$ results the most e cient scheme (one transmission per frame at a high MCS), leaving a lot of wireless resources to data stations. In contrast, with the legacy service there is only one transmission per frame, but data throughput is degraded due to the performance

[3] Additional experiments con rm that the performance with 24 Mb/s MCS is qualitatively similar.

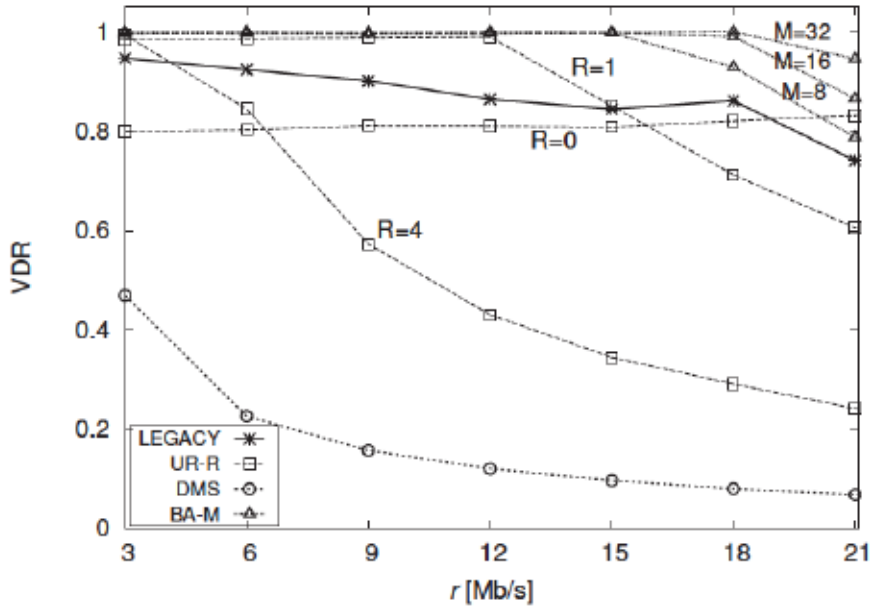


Figure 5.3: Performance of the mechanisms with synthetic traffic in channel 14. Successful video delivery ratio for different traffic loads.

anomaly (by increasing r the legacy traffic occupies longer the medium and, consequently, the overall throughput decreases). With $R = 1$, UR provides the same ADT as the legacy for $r \leq 12$ Mb/s, as two transmissions from the video station at the 54 Mb/s MCS consume almost the same amount of resources as one transmission at 24 Mb/s. The figure also illustrates the impact of the M parameter of BA, which serves to improve reliability as seen before, and also leaves more resources to data traffic but at the cost of increasing their latency. Finally, DMS provides a constant ADT that is higher than many schemes depending on the value of r. The reason is that, with this mechanism, the video transmitter performs a Binary Exponential Backoff (BEB) when transmission fails, which leaves more resources for data stations (for the other schemes, the transmitter never performs the BEB, acting more aggressively).

The above assessment is performed using channel 14, which is relatively unpopulated in our testbed. In order to understand the impact of harsher channel conditions on performance, we repeat the same experiments using channel 11, where we detect a number of WLANs with significant activity. We focus on the VDR, with the results depicted in Figure 5.5. According to the figure, the performance of the schemes is qualitatively very similar, although in most cases results are worse due to the increased interference, e.g., UR with $R = 1$ fails to guarantee video delivery. Only the BA scheme is able to keep its good performance.

Next, we perform another round of experiments in a more heterogeneous scenario with 10 video receivers, one of them suffering from very poor channel conditions. To enable

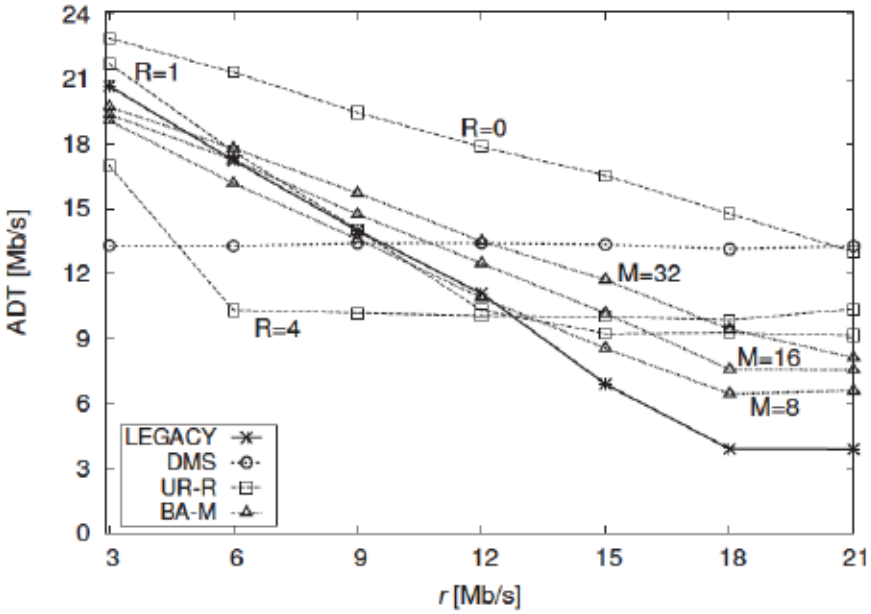


Figure 5.4: Performance of the mechanisms with synthetic traffic in channel 14. Aggregated data throughput for different traffic loads.

repeatability of the results, we emulate poor channel conditions by randomly discarding (with probability p=0.25) frames arriving at the impaired node. We measure the VDR for the nine stations with good channel conditions and for the receiver with poor reception, and depict the results in Figure 5.6. In the top subplot we present the results of the mean VDR for the stations with good channel conditions, and in the bottom subplot we depict the mean VDR of the station that randomly discards frames. As results illustrate, the GATS mechanisms maintain their relative good performance also in this new scenario for all the stations regardless the channel conditions; for instance, the BA mechanism with $M = 32$ is able to guarantee the video delivery for the impaired station for up to 12 Mb/s. Meanwhile, for this impaired station, the legacy scheme (and UR with $R = 0$) fails to provide 55 % delivery ratio even with 3 Mb/s, while the use of UR with $R = 4$, up to a load of 3 Mb/s, or the BA scheme significantly improves the delivery ratio.

5.2.3 Performance impairments

Here our interest is to quantify how efficient is our implementation, i.e., the difference between experimental results and the maximum achievable performance. To this aim, we use a backlogged desktop machine as AP and measure the maximum achievable bandwidth using GCR BA, with a packet length of $L = 1400$ B, to a group of 10 stations. Given that the 2.4 GHz band is quite populated in our testbed, we perform experiments during night on channel 14 where activity from external sources is negligible: for this reason links are very good and even if we fix data-rate to 54 Mb/s, with power set to 20 dBm we have

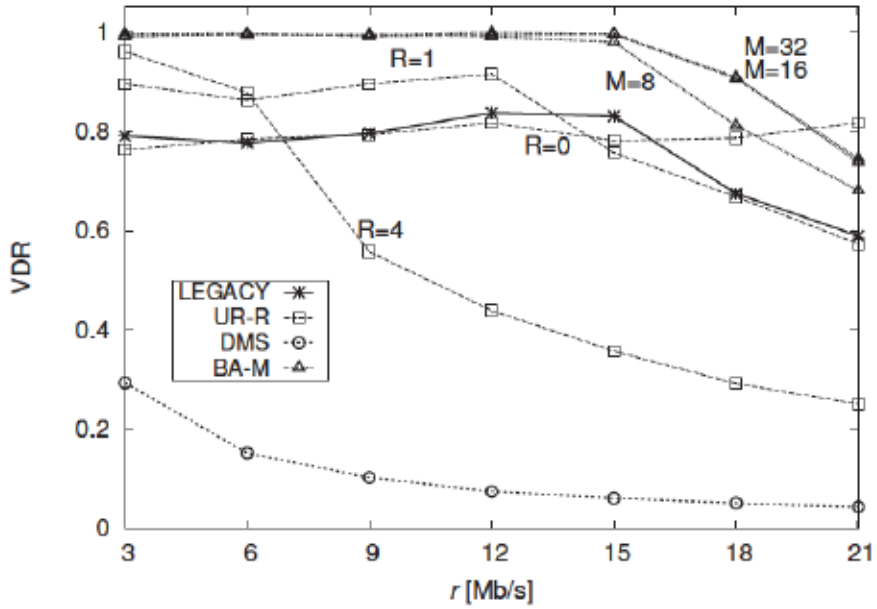


Figure 5.5: Successful video delivery ratio for different traffic loads in channel 11.

Table 5.2: Efficiency of the implementation

M	Theoretical R_i (Mb/s)	Experimental R_e (Mb/s)	η
8	30.81	24.25	78.7%
16	36.31	27.08	74.6%
32	39.86	28.61	71.8%

practically no losses.

We define the efficiency as $\eta = R_e/R_i$, with R_e being the experimental throughput (average of five 30-s experiments), and R_i the maximum theoretical one. This is computed as:

$$R_i = \frac{M \times L}{T_{\text{GCR BA Exchange}}} \tag{5.1}$$

where M is the size of the burst, L is the payload in bytes and $T_{\text{GCR BA Exchange}}$ is the total time for the exchange illustrated in Figure 5.7, assuming that the time to flush the queues and fetch new data is zero. We report the obtained values of R_e, R_i and η in Table 5.2, for different values of the burst length M (the higher the M the larger the throughput, as the relative amount of time spent in the BAREQ-BAREP pairs is smaller).

According to the results, it is clear that there is a notable efficiency loss, with all η values being below 80%. Furthermore, the higher the M the lower the η, which suggests that the reason for this performance impairment is that the chipset is not very efficient when handling long batches of frames. To understand this, we modify the firmware to produce verbose log files, identifying that the flushing and fetching operations take

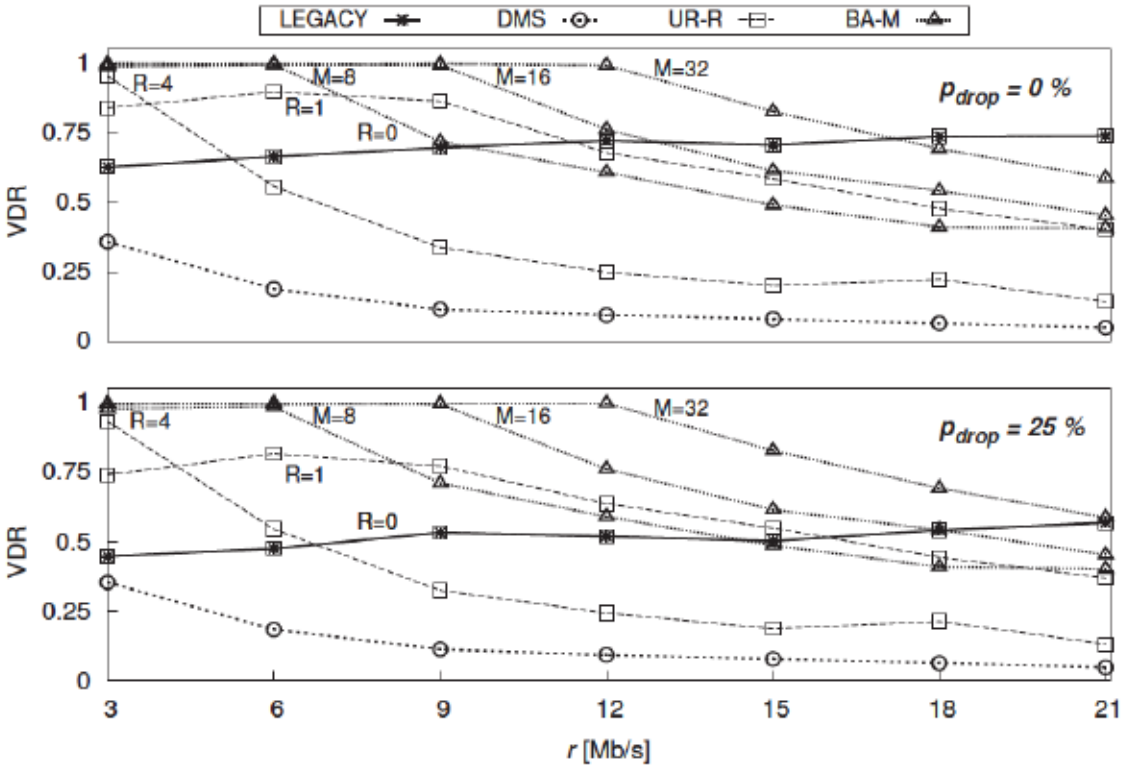


Figure 5.6: Successful video delivery ratio for different traffic loads in channel 11 with a station with $p_{drop} = 0.25$.

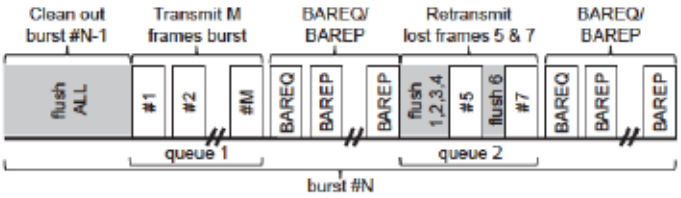


Figure 5.7: Basic Operation of GCR BA from the transmitter side.

relatively long periods of time. The reason for this is that these operations require the Direct Memory Access (DMA) system to operate with frames, which does not result very efficient in a number of circumstances, as we analyze next.

5.2.3.1 Anatomy of the bottleneck

To understand if the DMA operations are the main reason for the identified inefficiency, we set up the same scenario as in the previous experiments. However, in this case we tweaked the firmware in the AP to assume that all BAREPs report a successful reception of frames, which corresponds to the most demanding case in terms of DMA operations. Indeed, under this assumption, after the reception of the last BAREP the AP has to

empty all the retransmission queues and prepare for the next transmission burst, while in case some of the BAREP reported unsuccessful reception, the AP has to periodically switch between retransmission and ushing.

We also tweaked the rmware to measure the time t_f when the rst ush operation is made (right after the arrival of the last BAREP), and the time t_b when the rst frame of the next burst is available. These times are measured inside the NIC, thus avoiding the latency that appears when measuring inside the kernel due to the communication bus. We perform the experiment for three di erent burst sizes, $M =$ 8 16 32 , and packet lengths ranging from 100 B to 1400 B in steps of 100 B. For each of these con gurations, we perform ve 30 s experiments, collecting approximately 37500 time delays (i.e., t_b t_f). Then, in order to take into account the time required by ushing and fetching operations, we compute R_i adding the *median* of the time delays[4] to the denominator of (5.1). We reported the obtained values in Figure 5.8 as Experimental , while the values without this extra time are labeled as Theoretical .

According to the results, it is evident that the ushing and fetching of new data leads to substantial performance impairments, which become more drastic with larger values of M or L. Also, we note that the values for $L = 1400$ B correspond to the ones reported in Table 4.1, which validates to some extent our implementation of GCR BA, as the performance drop (for this value of L) can be attributed exclusively to the DMA operations, thus excluding any implementation issue.

We next analyzed in more detail the impact of M and L on the time to perform the ush and fetch operations. To this aim, we wrote an ad-hoc rmware that basically switches between two states: a *waiting state*, in which it does nothing while the application in user-space lls the DMA queue, and a *ushing state*, in which it records a timestamp, ushes the rst M packets from the queue, and records another timestamp, so the di erence between timestamps corresponds to the ushing time . We plot the resulting ushing times for representative values of M and L in Figure 5.9.

The gure illustrates that ushing times do not depend on the packet length until a certain threshold value L_{th}, and then grow linearly with L. This threshold depends on the burst size: the higher the M, the lower the L_{th}. Taking into account that each packet piggybacks a rmware header of 110 B, for all M cases with M $(L_{th} + 110)$ 3072 B, we have that the ushing times are constant (and very similar) for those con gurations in which the packet bursts ts in the internal memory of the NIC. Actually, according to our measurements when $L < L_{th}$ the ushing time amounts to a couple of us per packet, which is approximately the time required to execute the drop loop code. When $L > L_{th}$, new packets must be fetched from the main system memory, thus resulting in a linear growth of the ushing time with the quantity of transferred data. Given that

[4]We use the median of the collected samples as this removes spurious values due to clock wrap arounds.

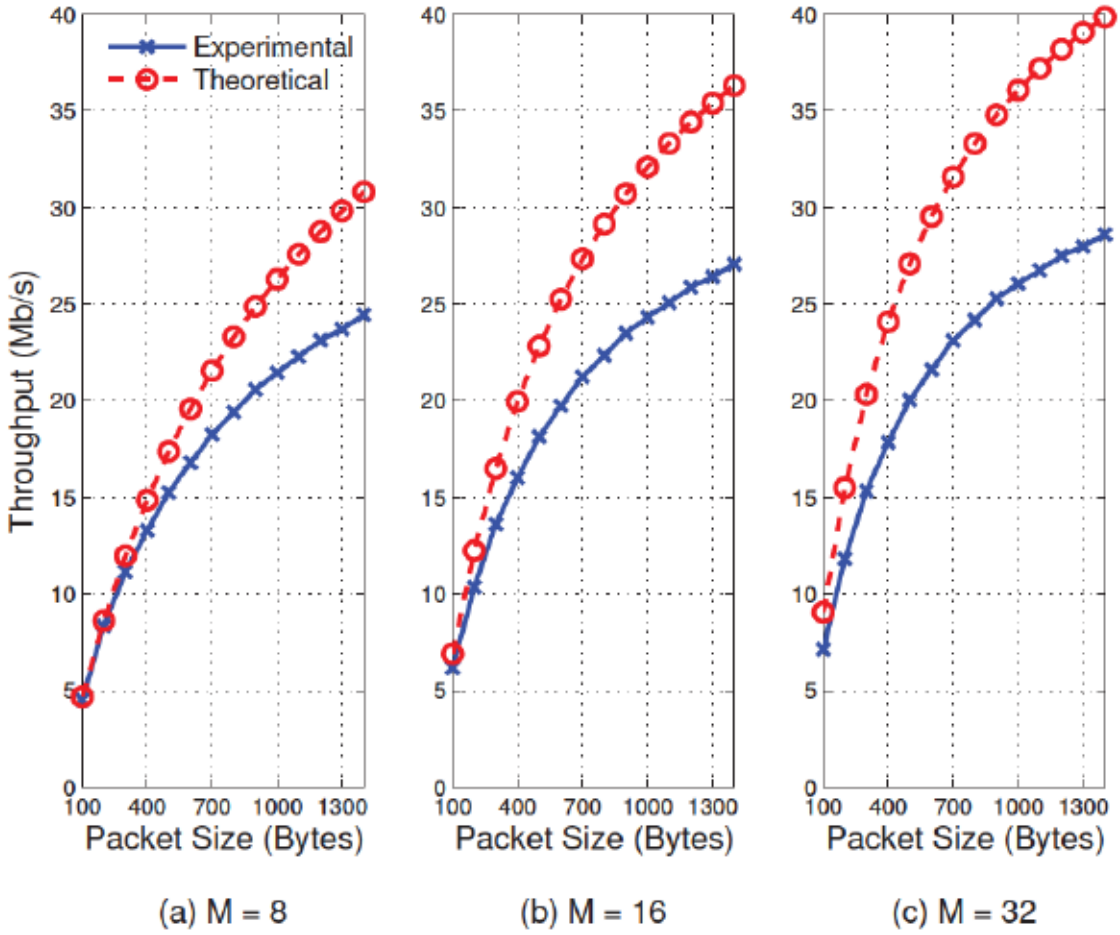


Figure 5.8: Ideal and experimental throughput (MCS = 54 Mb/s) of 802.11aa GCR BA for different burst sizes and different packet lengths.

the transfer rate we measured under these conditions (i.e., 44 Mb/s) is well below the theoretical capabilities advertised for mini-PCI devices (266 Mb/s), we conclude that the slow transfer rate not only depends on the overhead of the PCI signaling, but also on the DMA interface of the NIC which is not optimized. Still, it should be noted that the measured transfer rate of 44 Mb/s is well above the maximum achievable throughput with 802.11a/g, which is around 30 Mb/s.

In Appendix A we provide new directions for the revision of the current hardware architecture and propose a new vision for the future design of wireless chipsets to overcome the current limitations.

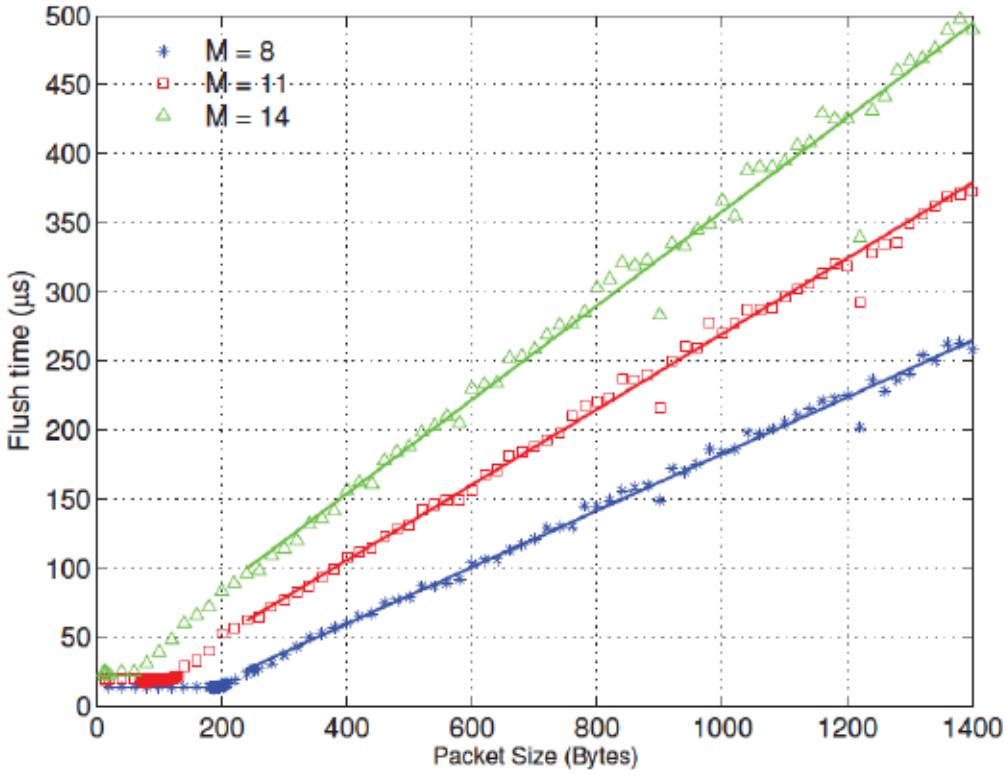


Figure 5.9: Flushing queue times for different burst sizes (M).

5.2.4 Real video

We next analyze the performance with real video traffic. To this aim, we stream one minute of the well-known "Big Buck Bunny" video[5] in Full HD (1920x1080) encoded with AVI MPEG4, resulting in an average bitrate of 12 Mb/s. To assess the impact of a high and low number of receivers and data stations, we consider two different cases for both N_v and N_d, namely, 5 and 15, leading to a total number of four different scenarios. We consider an additional value $M = 4$ to understand the impact of this parameter. We plot the resulting VDR[6] vs. ADT for each scenario in Figure 5.10.

For the case of BA, the results confirm its good properties, as in most cases its performance is around the top right corner of the figures, i.e., both high VDR and ADT. However, the results also highlight the need to adequately configure the parameter M (the burst length), as overly small values (i.e., $M = 4$) can on the one hand unnecessarily harm the performance of data stations even when the total number of stations is small ($\{N_v = 5, N_d = 5\}$), and on the other hand fail to guarantee the delivery of video when either the number of receivers or the data activity on the WLAN is high (i.e., $N_v = 15$

[5] `http://www.bigbuckbunny.org`

[6] Although in some cases we could compute the "quality" of the received video with, e.g., the peak signal-to-nose ratio, in other cases the resulting VDR is too low for the software tool to properly estimate it, which would have precluded a proper comparison of the multicast schemes.

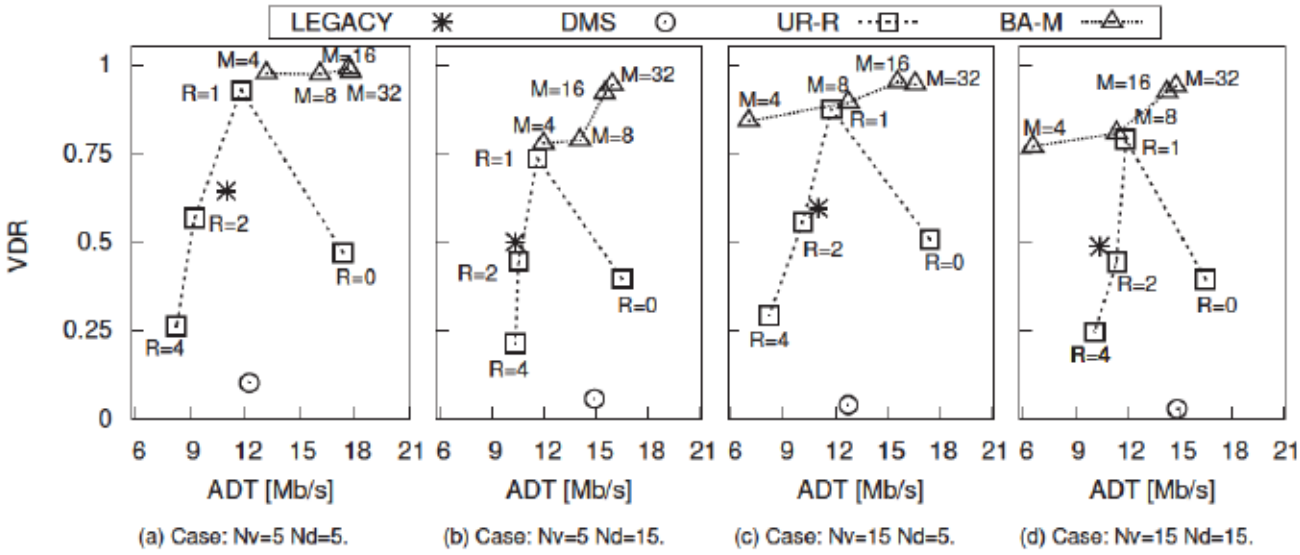


Figure 5.10: Received video file size and aggregated data throughput for the studied schemes.

or $N_d = 15$, respectively). Indeed, a poorly configured BA scheme can be outperformed by the much simpler UR scheme, if adequately configured, when the number of receivers is high (with $N_v = 15$, $M = 4$ vs. $R = 1$). Similarly, the need to limit the number of retransmissions with UR is clear from the figures: configuring UR with $R \geq 2$ results in a worse performance than the legacy scheme in terms of VDR, and similar performance in terms of ADT. Finally, the poor scalability of DMS is confirmed in this bandwidth-hungry scenario, with VDR values well below 20% even for small number of stations.

5.3 Summary

In this chapter, we have provided a first implementation and evaluation of the multicast mechanisms that are now available with 802.11aa. We have described how to prototype GATS using COTS hardware, made our implementation available and we have performed an extensive experimental assessment in a variety of scenarios. As we previewed in the previous chapter where we introduced an initial evaluation of 802.11aa our prototype shows that GATS significantly improves the performance of video delivery over 802.11 WLANs, and is implementable over existing devices, unlike previous multicast proposals relying on non-standard or complex functionality.

Results confirm that each mechanism offers a specific trade-off between complexity and performance (Table 5.3). Our experimentation has considered a large variety of scenarios for different settings of R (number of retries in GCR UR), and M (burst length in GCR

Table 5.3: Assessment of the multicast schemes

Scheme	Complexity	E ectiveness	E ciency
Legacy	None	Medium	Low
UR	Medium	Medium-High	Medium
DMS	Medium	None	Medium
BA	High	High	High

BA), leading to very di erent performance in terms of throughput and video delivery.

Chapter 6

Conclusions and Future Work

Voice and video tra c represent an important part of total Internet tra c and 802.11 is the most widely used wireless technology to gain Internet access. Several techniques have been applied to improve the performance of these kinds of tra c. While new amendments have prioritized the voice and video tra c, in the case of voice they still do not solve its huge overhead, in terms of control bytes and backo procedure. This thesis lls this gap by proposing a novel MAC mechanism, *VoIPiggy*, which extends the control frames sent from the stations to the AP with user data. VoIPiggy adapts to the network conditions and is transparent to the application layer. This mechanism provides a fair access to the medium to all the voice clients and frees resources for the data stations. We provide an analytical model to predict its performance, in terms of capacity region and delay. Note that we perform our implementation over commodity hardware, unlike previous proposals relying on non-standard or complex functionality. In addition, experimental - with up to 30 nodes - and analytical results evidence that our solution doubles the voice capacity in a WLAN scenario. A future line of work is the implementation of a Call Admission Control (CAC), leveraging the adaptability of VoIPiggy and the available resources in the WLAN freed by a more e cient voice transmission. Additionally, we nd interesting to integrate VoIPiggy within standard mechanisms, for instance the reverse direction mechanism of 802.11n.

For the case of video multicast streaming, WLANs lack a feedback mechanism and transmit at a lower MCS, causing the performance anomaly problem. 802.11aa has been proposed to improve the delivery of multicast video streams. However, the novelty of the standard and the di culty in hardware changes have dismissed its implementation. Furthermore, the standard leaves open the con guration of the parameters of its mechanisms. We perform a thorough performance evaluation of the di erent mechanisms of 802.11aa laying the foundations for choosing the appropriate mechanism according to the network conditions. Simulation results proved the e ective behavior of GATS mechanisms in an extensive set of scenarios. Furthermore, we develop the rst implementation of the GATS

mechanisms and release open source to the research community. Our experimentation has considered a large variety of scenarios, and opens a number of research questions that need to be tackled in the future. In conclusion, no scheme is declared the best, since the performance deeply varies depending on the size of the multicast group as well as on the di erent GATS parameters: except for the legacy service, they all can be optimal in a given scenario. In this way, one rst research challenge to tackle is their optimal con guration, given some performance criterion. Another problem is the design of the best policy to program the Block Ack (e.g., when to hold frames, how to schedule the polling of stations, the setting of the MSDU lifetime parameter). We think that our contribution, as we release the rst open source implementation of 802.11aa, will have a signi cant impact in the adoption of future standards, such as 802.11ax, which builds on top of 802.11aa.

Given that the 802.11aa standard leaves a lot of room for implementation dependent optimizations (for these and other questions), and based on our previous experiences, we foresee that 802.11aa hardware will have either hard-coded mechanisms whose functionality cannot be altered, or some extensions but poorly documented (at most). By making our implementation publicly available, we hope to foster researchers and practitioners to prototype and assess their optimizations in the above or other areas. Thus, as a future direction of this line of work, we study the possibility of experimentally perform a more through validation considering a wider variety of video qualities and also di erent type of data tra c.

References

[1] *IEEE Standard for Information Technology-Telecommunications and information exchange between systems-Local and metropolitan area networks-Speci c requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) speci cations*, IEEE Std. 802.11, 2012.

[2] Cisco Visual Networking Index: Global Mobile Data Tra c Forecast Update, 2014-2019, White paper, Cisco, Feb. 2015. [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.pdf

[3] Cisco Visual Networking Index: Forecast and Methodology, 2014-2019, White paper, Cisco, May 2015. [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/ip-ngn-ip-next-generation-network/white_paper_c11-481360.pdf

[4] *IEEE P802.11ac/D4.1 Draft Standard - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Speci cations - 4: Enhancements for Very High Throughput for Operation in Bands below 6 GHz*, IEEE Amendment 802.11ac, 2012.

[5] *IEEE P802.11ad/D9.00 Draft Standard - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Speci cations - Amendment 3: Enhancements for Very High Throughput in the 60 GHz Band*, IEEE Amendment 802.11ad, 2012.

[6] *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Speci cations - Amendment 8: Medium Access Control (MAC) Quality of Service Enhancements*, IEEE Amendment 802.11e, 2005.

[7] A. Banchs, P. Serrano, and L. Vollero, Reducing the Impact of Legacy Stations on Voice Tra c in 802.11e EDCA WLANs, *IEEE Communications Letters*, vol. 11, no. 4, April 2007.

[8] J. Kuri and S. Kasera, Reliable Multicast in Multi-access Wireless LANs, in *Proceedings of IEEE INFOCOM*, vol. 2, Mar. 1999, pp. 760 767 vol.2.

[9] P. Serrano, A. Banchs, P. Patras and A. Azcorra, Optimal Con guration of 802.11e EDCA for Real-Time and Data Tra c, *IEEE Transactions on Vehicular Technology*, vol. 59, no. 5, pp. 2511 2528, Jun. 2010.

[10] *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Speci cations - Amendment 3: MAC Enhancements for Robust Audio Video Streaming*, IEEE Amendment 802.11aa, 2012.

[11] P. Salvador, F. Gringoli, V. Mancuso, P. Serrano, A. Mannocci, and A. Banchs, VoIPiggy: Implementation and evaluation of a mechanism to boost voice capacity in 802.11 WLANs, in *Proceedings of IEEE INFOCOM*, March 2012, pp. 2931 2935.

[12] P. Salvador, V. Mancuso, P. Serrano, F. Gringoli, and A. Banchs, VoIPiggy: Analysis and Implementation of a Mechanism to Boost Capacity in IEEE 802.11 WLANs Carrying VoIP Tra c, *IEEE Transactions on Mobile Computing*, vol. 13, no. 7, pp. 1640 1652, July 2014.

[13] A. De La Oliva, P. Serrano, P. Salvador, and A. Banchs, Performance Evaluation of the IEEE 802.11aa Multicast Mechanisms for Video Streaming, in *Proceedings of IEEE WoWMoM*, June 2013, pp. 1 9.

[14] P. Serrano, P. Salvador, V. Mancuso, and Y. Grunenberger, Experimenting With Commodity 802.11 Hardware: Overview and Future Directions, *IEEE Communications Surveys Tutorials*, vol. 17, no. 2, pp. 671 699, Second quarter 2015.

[15] P. Salvador, S. Paris, C. Pisa, P. Patras, Y. Grunenberger, X. Perez-Costa, and J. Gozdecki, A Modular, Flexible and Virtualizable Framework for IEEE 802.11, in *Future Network Mobile Summit (FutureNetw)*, July 2012, pp. 1 8.

[16] P. Salvador, L. Cominardi, F. Gringoli, and P. Serrano, A First Implementation and Evaluation of the IEEE 802.11aa Group Addressed Transmission Service, *ACM SIGCOMM Computer Communications Review*, vol. 42, no. 1, pp. 13 18, 2014.

[17] P. Salvador, F. Gringoli, P. Serrano, N. Facchi, and S. Paris, Making a Case for Flexible 802.11 Architectures, in *Communications (ICC), 2015 IEEE International Conference on*, June 2015, pp. 3678 3684.

[18] C. Vlachou, A. Banchs, P. Salvador, J. Herzen, and P. Thiran, Analysis and Enhancement of CSMA/CA with Deferral in Power-Line Communications, *IEEE Journal on Selected Areas (JSAC)*, Accepted for publication.

[19] G. Hiertz, D. Denteneer, L. Stibor, Y. Zang, X. Perez-Costa, and B. Walke, The IEEE 802.11 Universe, *IEEE Communications Magazine*, vol. 48, no. 1, pp. 62 70, January 2010.

[20] *IEEE 802.1Qat/D6.1 Draft Standard for Information Technology-Telecommunications and information exchange between systems-Local and metropolitan area networks: Virtual Bridged Local Area Networks - Amendment 9: Stream Reservation Protocol (SRP)*, IEEE Std. 802.1Qat, 2010.

[21] S. Garg and M. Kappes, Can I add a VoIP call? in *Proceedings of IEEE ICC*, Anchorage, Alaska, USA, May 2003.

[22] S. Garg and M. Kappes, An Experimental Study of Throughput for UDP and VoIP Tra c in IEEE 802.11b Networks, in *Proceedings of WCNC*, New Orleans, LA, USA, March 2003.

[23] P. Serrano, A. Banchs, J. F. Kukielka, G. D Agostino, and S. Murphy, Con guration of IEEE 802.11e EDCA for Voice and Data tra c: An Experimental Study, in *Proceedings of ICT-MobileSummit*, Stockholm, Sweden, Jun 2008.

[24] G. Hanley, S. Murphy, and L. Murphy, Adapting WLAN MAC Parameters to Enhance VoIP Call Capacity, in *Proceedings of MSWiM*, October 2005, pp. 250 254.

[25] F. Anjum, M. Elaoud, D. Famolari, A. Ghosh, R. Vaidyanathan, A. Dutta, P. Agrawal, T. Kodama, and Y. Katsube, Voice Performance in WLAN Networks An Experimental Study, in *Proceedings of IEEE GLOBECOM*, December 2003.

[26] *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Speci - cations - Amendment 5: Enhancements for Higher Throughput*, IEEE Amendment 802.11n, 2009.

[27] M. Ozdemir, D. Gu, A. McDonald, and J. Zhang, Enhancing MAC Performance with a Reverse Direction Protocol for High-Capacity Wireless LANs, in *Proceedings of IEEE VTC*, September 2006.

[28] Hsiang-Ho Lin and Chih-Yu Wang and Hung-Yu Wei, Improving Online Game Performance over IEEE 802.11n Networks, in *Proceedings of NetGames*, November 2010.

[29] M. Rashid, E. Hossain, and V. Bhargava, HCCA Scheduler Design for Guaranteed QoS in IEEE 802.11e Based WLANs, in *Proceedings of IEEE WCNC*, 2007, pp. 1538 1543.

[30] H.-J. Lee, J.-H. Kim, and S. Cho, A Novel Piggyback Selection Scheme in IEEE 802.11e HCCA, in *Proceedings of IEEE ICC*, 2007.

[31] C. Casetti, C. F. Chiasserini, M. Fiore, and M. Garetto, Notes on the Ine ciency of 802.11e HCCA, in *Proceedings of IEEE VTC*, 2005.

[32] I. Tinnirello, G. Bianchi, P. Gallo, D. Garlisi, F. Giuliano, and F. Gringoli, Wireless MAC Processors: Programming MAC Protocols on Commodity Hardware, in *Proceedings of IEEE INFOCOM*, March 2012, pp. 1269 1277.

[33] Lee, Hyun-Jin and Kim, Jae-Hyun and Cho, Sunghyun, A Novel Piggyback Selection Scheme in IEEE 802.11e HCCA, in *Proceedings of IEEE ICC*. IEEE, 2007, pp. 4529 4534.

[34] Y. Xiao, IEEE 802.11 Performance Enhancement via Concatenation and Piggyback Mechanisms, *IEEE Transactions on Wireless Communications*, vol. 4, no. 5, pp. 2182 2192, September 2005.

[35] P. Verkaik, Y. Agarwal, R. Gupta, and A. C. Snoeren, Softspeak: Making VoIP Play Well in Existing 802.11 Deployments, in *Proceedings of NSDI*, 2009, pp. 409 422.

[36] O. Alay, T. Korakis, Y. Wang, and S. Panwar, Dynamic Rate and FEC Adaptation for Video Multicast in Multi-rate Wireless Networks, *Mobile Network Applications*, vol. 15, no. 3, pp. 425 434, Jun. 2010.

[37] C. Xi, W. YunHeng and L. JianHua, Cross-layer Link Rate Adaptation for High Performance Multimedia Broadcast over WLANs, in *Proceedings of IEEE GLOBECOM Workshops (GC Wkshps)*, Dec. 2010, pp. 965 969.

[38] A. Lyakhov, V. Vishnevsky and M. Yakimov, Multicast QoS Support in IEEE 802.11 WLANs, in *Proceedings of IEEE MASS, 8-11 Oct. 2007, Pisa, Italy.* IEEE, 2007.

[39] M.-T. Sun, L. Huang, A. Arora, and T.-H. Lai, Reliable MAC Layer Multicast in IEEE 802.11 Wireless Networks, in *Proceedings of the 2002 International Conference on Parallel Processing*, ser. ICPP 02. Washington, DC, USA: IEEE Computer Society, 2002, pp. 527 .

[40] A. Lyakhov and M. Yakimov, Analytical Study of QoS-Oriented Multicast in Wireless Networks, *EURASIP Journal Wireless Communications and Networking*, vol. 2011, 2011.

[41] J. Miroll, Z. Li, and T. Herfet, Wireless Feedback Cancellation for Leader-Based MAC Layer Multicast Protocols, in *Proceedings of IEEE ISCE*, June 2010, pp. 1 6.

[42] S. Choi, N. Choi, Y. Seok, T. Kwon, and Y. Choi, Leader-Based Rate Adaptive Multicasting for Wireless LANs, in *Proceedings of IEEE GLOBECOM*, Nov. 2007.

[43] J. Miroll and T. Herfet, E cient OFDM-based WLAN-multicast with feedback aggregation, power control and rate adaptation, *Computer Communications*, vol. 35, no. 18, pp. 2245 2253, 2012.

[44] S. Gupta, V. Shankar, and S. Lalwani, Reliable Multicast MAC Protocol for Wireless LANs, in *Proceedings of IEEE ICC*, vol. 1, May 2003, pp. 93 97.

[45] Zhengyong Feng and Guangjun Wen and Chunlei Yin and Hongsheng Liu, Video Stream Groupcast Optimization in WLAN, in *International Conference on Internet Technology and Applications*, 2010.

[46] K. Maraslis, P. Chatzimisios, and A. Boucouvalas, 802.11aa: Improvements on Video Transmission over Wireless LANs, in *Proceedings of IEEE ICC*, Jun. 2012.

[47] M. A. Santos, J. Villalon, and L. Orozco-Barbosa, Evaluation of the IEEE 802.11aa Group Addressed Service for Robust Audio-Video Streaming, in *Proceedings of IEEE ICC*, Jun. 2012, pp. 6879 6884.

[48] S. A. Baset and H. G. Schulzrinne, An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol, in *Proceedings of IEEE INFOCOM*, April 2006.

[49] Y.-C. Chang, K.-T. Chen, C.-C. Wu, and C.-L. Lei, Inferring Speech Activity from Encrypted Skype Tra c, in *Proceedings of IEEE GLOBECOM*, 2008.

[50] R. Ramjee, J. Kurose, D. Towsley, and H. Schulzrinne, Adaptive Playout Mechanisms for Packetized Audio Applications in Wide-Area Networks, in *Proceedings of IEEE INFOCOM*, June 1994, pp. 680 688.

[51] G. Bianchi, Performance Analysis of the IEEE 802.11 Distributed Coordination Function, *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 3, pp. 535 547, March 2000.

[52] A. Eckberg Jr., The Single Server Queue with Periodic Arrival Process and Deterministic Service Times, *IEEE Transactions on Communications*, vol. 27, no. 3, pp. 556 562, Mar 1979.

[53] A. Kashyap, U. Paul, and S. R. Das, Deconstructing Interference Relations in WiFi Networks, in *Proceedings of IEEE SECON*, June 2010.

[54] B. Han, A. Schulman, F. Gringoli, N. Spring, B. Bhattacharjee, L. Nava, L. Ji, S. Lee, and R. Miller, Maranello: Practical Partial Packet Recovery for 802.11, in *Proceedings of NSDI*, March 2010, pp. 205 218.

[55] P. Gallo, F. Gringoli, and I. Tinnirello, On the Flexibility of the IEEE 802.11 Technology: Challenges and Directions, in *Proceedings of the Future Network & Mobile Summit 11*, Poland, June 2011.

[56] *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications - Amendment 8: IEEE 802.11 Wireless Network Management*, IEEE Amendment 802.11v, 2011.

[57] ITU-T Recommendation H.264 : Advanced video coding for generic audiovisual services, International Telecommunications Union, Nov. 2007.

[58] P. Seeling and M. Reisslein, Video Transport Evaluation With H.264 Video Traces, *IEEE Communications Surveys and Tutorials*, vol. 14, no. 4, pp. 1142 1165, 2012.

[59] G. Pei and T.-R. Henderson., Validation of the OFDM error rate model in ns-3, 2010.

[60] M. Garetto, T. Salonidis and E.-W. Knightly, Modeling Per- ow Throughput and Capturing Starvation in CSMA Multi-hop Wireless Networks, *IEEE/ACM Transactions on Networking*, vol. 16, no. 4, Aug. 2008.

[61] Y. Liang, J. Apostolopoulos, and B. Girod, Analysis of Packet Loss for Compressed Video: E ect of Burst Losses and Correlation Between Error Frames, *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 7, pp. 861 874, Jul. 2008.

[62] S. Wenger, Common conditions for wire-line, low delay. IP/UDP/RTP packet loss resilient testing, in *ITU-T VCEG document VCEG-N79r1*, Sep. 2001.

[63] U. Horn, K. Stuhlmuller, M. Link and B. Girod, Robust Internet Video Transmission Based on Scalable Coding and Unequal Error Protection, vol. 15, 1999, pp. 77 94.

[64] T. Stockhammer, M.-M. Hannuksela and T. Wiegand, H.264/AVC in Wireless Environments, *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, pp. 657 673, 2003.

[65] T. Schierl, H. Schwarz, D. Marpe, and T. Wiegand, Wireless Broadcasting Using the Scalable Extension of H.264/AVC, in *ICME*. IEEE, 2005, pp. 884 887.

[66] P. Salvador, L. Cominardi, F. Gringoli, and P. Serrano, Technical Report: Implementations details of the IEEE 802.11aa Group Addressed Transmission Service, pp. 1 25, August 2013.

[67] A. Ganz, A. Savvides, and Z. Ganz, Media Access Control Development Platform for Wireless LANs, in *Proceedings of IEEE ICECS*, 1999.

[68] A. Sharma and E. Belding, FreeMAC: Framework for Multi-Channel MAC Development on 802.11 Hardware, in *Proceedings of ACM PRESTO*, 2008.

[69] M. Neufeld, J. Fi eld, C. Doerr, A. Sheth, and D. Grunwald, SoftMAC: A Flexible Wireless Research Platform, in *Proceedings of ACM HotNets*, 2005.

[70] L. Wisniewski, H. Trsek, I. Dominguez-Jaimes, A. Nagy, R. Exel, and N. Kero, Location-based Handover in Cellular IEEE 802.11 Networks for Factory Automation, in *Proceedings of IEEE EFTA*, 2010.

[71] K. Tan, J. Zhang, J. Fang, H. Liu, Y. Ye, S. Wang, Y. Zhang, H. Wu, W. Wang, and G. M. Voelker, Sora: High Performance Software Radio Using General Purpose Multi-core Processors, in *Proceedings of USENIX NSDI*, 2009, pp. 75 90.

[72] A. Khattab, J. Camp, C. Hunter, P. Murphy, A. Sabharwal, and E. W. Knightly, WARP: A Flexible Platform for Clean-slate Wireless Medium Access Protocol Design, *SIGMOBILE Mobile Computer Communications Review*, vol. 12, no. 1, pp. 56 58, 2008.

[73] D. Armstrong and M. Pearson, A Rapid Prototyping Platform for Wireless Medium Access Control Protocols, in *Proceedings of IEEE ASAP*, 2007, pp. 403 408.

[74] I. Tinnirello, G. Bianchi, P. Gallo, D. Garlisi, F. Giuliano, and F. Gringoli, Wireless MAC Processors: Programming MAC Protocols on Commodity Hardware, in *Proceedings of IEEE INFOCOM*, 2012, pp. 1269 1277.

[75] G. Bianchi, P. Gallo, D. Garlisi, F. Giuliano, F. Gringoli, and I. Tinnirello, MAClets: Active MAC Protocols over Hard-Coded Devices, in *Proceedings of ACM CoNext*, 2012, pp. 229 240.

[76] Y. Dong, D. Xu, Y. Zhang, and G. Liao, Optimizing Network I/O Virtualization with E cient Interrupt Coalescing and Virtual Receive Side Scaling, in *Proceedings of IEEE CLUSTER*, 2011, pp. 26 34.

[77] H. Rauchfuss, T. Wild, and A. Herkersdorf, A Network Interface Card Architecture for I/O Virtualization in Embedded Systems, in *Proceedings of USENIX WIOV*, 2010, pp. 1 7.

Appendix A

Limitations of Flexibility in Current 802.11 Architectures and New Directions

In Chapter 5 we have demonstrated the lack of exibility in the way that NIC modules interact resulting in a bottleneck that severely degrades performance. Here, we provide new directions for the revision of the current hardware architecture and propose a new vision for the future design of wireless chipsets.

A.1 Hardware support and limitations

The seminal work of [67] was among the rst to underline how commercial NICs could be turned into exible research platforms, by replacing the stock rmware with a di erent one o ering custom transmission/reception paths. However, even if this work revealed the exibility of the PRISM 11b chipset at the MAC layer with its MAC CPU, it did not disclose how to modify the rmware, which remained closed-source. Since then, and with the end of the PRISM project,[1] many works selected Atheros-based NICs as the next research platform of choice, thanks to the active collaboration of the manufacturer, excellent open-source drivers (e.g.: Mad-Wi) and a lot of documentation. Apart from works targeting speci c features, [68] and [69] provided software frameworks for the easy customization of some functionalities of the Atheros AR5212 chipset, such as disabling packet acknowledgment, fast channel switching and modifying the number of retries for failed frames.

None of the above supported ad-hoc frame exchanges with the required timing, e.g., receiving a frame and forging the corresponding reply within a few microseconds. Because of this, a number of projects designed 802.11-compliant implementations from scratch, many of them building around an FPGA: e.g., SMiLE [70], used to test novel localization schemes; or Sora [71] and WARP [72], the former being a very powerful Software De ned Radio (SDR) device that connects to the computer that handles frame-symbol transformation through a mini-PCI interface, and the latter

[1] The maintenance of the PRISM project ended up in 2007.

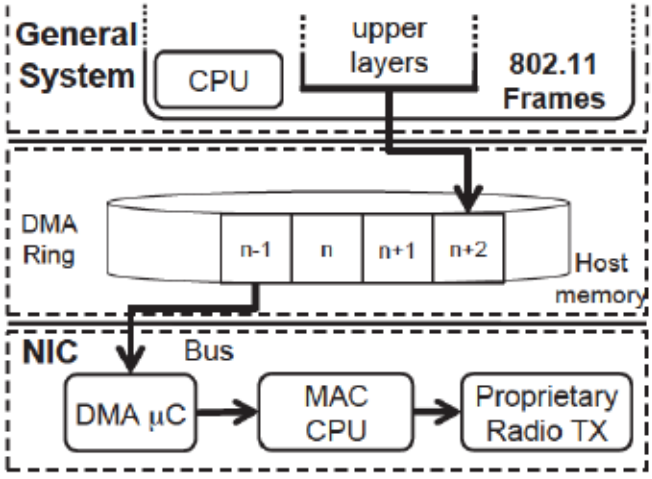


Figure A.1: NIC high level architecture: transmission path.

being a standalone device that executes upper and lower MAC functions in two different soft-cores. A platform really similar to a real WiFi NIC, i.e., with a MAC CPU synthesised on a FPGA executing the MAC firmware and a mini-PCI interface for connecting to the main host, was introduced in [73]: unfortunately, the project did not evolve further from the 802.11b stage.

Differently than all previous approaches, the architecture proposed in the European project FLAVIA[2] decoupled the logic that drives an access protocol from the hardware implementation. Authors demonstrated this vision with the Wireless MAC Processor on top of inexpensive consumer NICs from Broadcom, showing the possibility to re-program the MAC behaviour in real-time with a graphical oriented programming language [74, 75].

To the best of our knowledge, no previous work has identified the need to bring flexibility also at the *frame interface* between the host and the wireless NIC. This could look rather surprising, given that this sort of flexibility was introduced in the last few years to enable a faster bottleneck-free communication between the host and some wired interfaces like with the Receive-Side Scaling technique used by Intel on 10Gb Ethernet NIC [76, 77].

A.2 Current hardware designs

WiFi NICs are typically based on the hardware architecture depicted in Figure A.1, which details the transmission path (the process on the reception path is basically the reciprocal case). In contrast to old-styled *Programmed Input/Output* (PIO) designs, modern approaches use DMA for transferring data between the memory of the main host running the OS and the NIC circuitry. A kernel thread first writes frames into a consecutive set of memory pages, which are arranged in a circular ring of DMA slots, postponing the operation if there are no free slots; then it programs the DMA controller on the NIC to start retrieving and transmitting frames.

On the NIC side, the MAC CPU focuses on re/transmission operations, with the host communication being offloaded by the DMA subsystem that caches the first few frames from every

[2]FP7 ICT FLAVIA project,"FLexible Architecture for Virtualizable future wireless Internet Access" `http://www.ict-flavia.eu`

queue in the NIC internal memory. When the *BC* of a queue reaches zero, the MAC CPU schedules the transmission of the Head-of-Line frame and when the frame is acknowledged or exceeds the maximum retransmission attempts, the MAC CPU reports a *frame done* to the kernel via an interruption request (IRQ). Then, the corresponding DMA micro-controller (DMA C) starts fetching the next frame, and the kernel reclaims the DMA slot in the ring for the new frames that may arrive from upper layers.

Two are the main bene ts of this DMA-based approach: (*i*) there are always frames available to the NIC, unlike the PIO case; and (*ii*) the main host CPU is relieved while the NIC transmits frames. Both the host CPU and the NIC CPU focus on their very speci c tasks and frames can be transmitted at the maximum throughput allowed by the standard timings.

In nuce, this scheme emulates a simple FIFO queue for pushing frames to the NIC, and therefore it results adequate as long as there is a strict order in handling the frames: when the Head-of-Line (HOL) frame is served the NIC may access a new one.[3] However, for the case of Block Acknowledgment the operation is slightly di erent: not all the frames in a burst or an A-MPDU need to be retransmitted, as they might experience di erent channel conditions. Unfortunately, once the burst is handled, none of the frames are available at the NIC for retransmission, thereby this calls for the collaboration of the host kernel. After processing the BAREP, the kernel re-sends to the NIC the requested frames, possibly interleaved with new ones.[4] However, for 802.11aa this will slow down the video streaming rate or mix old frames, likely to have expired, with new ones. We describe in the following how we overcome this issue in our 802.11aa implementation, underlining how this version of BA (in contrast to the 802.11n) is hard to deploy with the current DMA-based operation.

A.3 Evolved Architecture for 802.11 platforms

Based on our experience reported in Section 5.2.3.1, we argue that apart from improving the mechanisms for transferring data between the NIC and the host system, the development of future wireless NICs should be (even more) inspired by current computer architectures. In this approach, *software developers* write both user programs and the OS, which interfaces to the hardware through a set of speci c drivers released by *manufacturers*. This maps into NIC internals to an architecture where MAC algorithms are developed by *protocol implementers* using APIs created by *manufacturers* for accessing the NIC hardware. We next describe in detail our technical vision and its advantages.

A.3.1 Technical description

We present in Figure A.2 the evolved NIC architecture, thereby we build on Figure A.1, and focus again on the transmission path. First, we introduce a *Real Time OS* executed by the NIC CPU that isolates the *MAC threads* from the mechanisms that exchange data with the host, and from the low level transmission primitives implemented by *proprietary PHY drivers*.

At the radio, we still discard the SDR approach even if *ultimate* choice for targeting PHY

[3]For the case of the multiple queues supported in EDCA, it can be extended with multiple DMA controllers, each programmed to present di erent memory blocks to the NIC.

[4]Note that this operation holds for current 802.11n chipsets, e.g., Broadcom 43224 and 43225 ones.

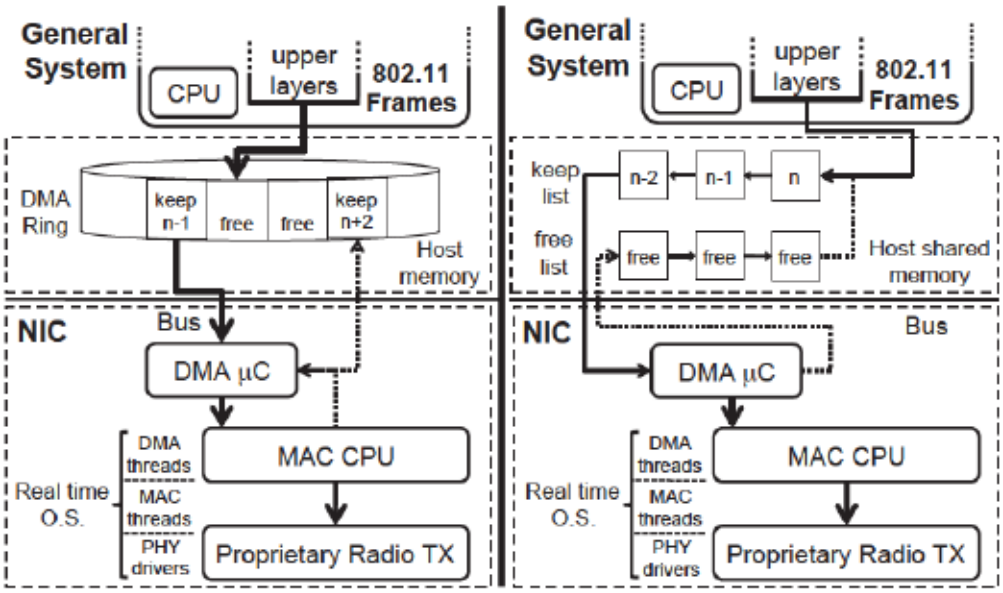


Figure A.2: Visionary architectures for 802.11 NICs.

flexibility: it either requires a really powerful internal CPU[5] or fails to provide decent MAC support if the host CPU is involved because of latencies. Also the FPGA solution is unfeasible: apart from energy issues, OFDM compliant PHY requires a "fat" FPGA with too many gates which makes it too expensive for consumer electronics. For these reasons passband-PHY circuitry should still be wired to inexpensive and energy efficient designs. This limits the flexibility of the radio, but manufacturers could easily strive PHYs to avoid strictly fixed MCS: e.g., they can allow reconfiguration of number of OFDM sub-carriers, ratio of the convolutional en/decoder for bit error protection, mapping between user-bits and symbols, and other parameters like the frequency and transmission power.

With regards to the *MAC CPU*, given that passband operations are offloaded by the hardwired PHY and considering the long MAC timings, even a slow clock of 44 MHz like in current chipsets[6] is enough to schedule frame transmission within sub-microsecond granularity. To this end the integration of an energy efficient ARM architecture, currently spreading as co-processors for many tasks (e.g., H.264 encoders), could be even more convenient than acquiring a third party design.

As for the communication mechanism between the host and the NIC, which we demonstrated in Section 5.2.3 to be the real bottleneck for the new GCR BA procedure, and showed to be the major concern for the actual implementation of many recent proposals, we propose three solutions. The first two build on improvements of the DMA system: here the main problem is the looping over a ring of slots using a producer/consumer paradigm, with the MAC CPU allowed to only *release* a slot after usage.

The MAC program could instead report to the host that the slot is either *released* or *kept*: only in the first case, the slot will be replaced with a new frame. Then, the MAC program loops again over *kept* frames, directly programming the DMA controller for locating specific slots (Figure A.2

[5]With USRP and GnuRadio, decoding 11g frames at highest MCS pushes full load on a modern x86 CPU

[6]Atheros chipsets are clocked at 44 MHz, Broadcom at 88 MHz

left). A second solution is to enable complete memory sharing between host and MAC CPU, as in inexpensive video chipsets, by means of a data structure (double linked list). The rst list, populated by the host with new frames, is drained by the MAC CPU that moves released frames into the second list, and the host CPU will recycle these frames (Figure A.2 right). This option requires simpler mechanisms at both host and NIC CPUs for atomic read and assignment, which are already present on inexpensive ARM architectures. These two solutions permit manufacturers to keep low NIC memory and costs. A third solution, not reported in the gure for simplicity, consists of increasing the internal memory, so that the DMA system is only used for copying frames for later internal use by the NIC.

A.3.2 Advantages

Similarly to what happened in the computer world, abandoning the monolithic approach in favor of our more structured proposal, which integrates the operation of components provided by di erent players, would lead to a number of advantages. First, *hardware manufacturers* can focus on what really di erentiates NICs, i.e.: the enhancement of key radio components, such as, the front-end to recover the clock and decode signals in noisy environments; the design of energy-e cient techniques, and the introduction of new mechanisms for de/encoding spatially multiplexed signal streams. Second, *protocol implementers* shall not have to deal with hardware speci c details, hidden by primitives for transmitting and receiving frames, scheduling timers or con guring PHY speci c parameters. Instead, they could focus on the actual protocol implementation, that would bene t from continuous software upgrades lined up with the latest version of a standard. Third, *developers of the Real time OS* specialize in improving algorithms for the allocation of the NIC internal resources to either MAC programs or driver threads; or on enhancing data communication mechanisms with the host by tuning the management of the shared memory. Fourth, *developers of the Host OS* do not have to implement di erent drivers, as they would solely interface with a *generic* device, leading to easier porting of the NIC itself to many OS. Finally, *consumers* will bene t from a better network experience, with resource-e cient devices that are constantly updated with new OS and protocol releases.

A.4 Summary

In this Appendix, we have illustrated that existing 802.11 interfaces are based on generic hardware architectures, which are exible enough to implement new mechanisms introduced by recent standards, but present certain bottlenecks that preclude an e cient operation. Building on our experiences from implementing the 802.11aa GATS mechanisms, we have proposed an evolved hardware architecture for these interfaces, inspired by the evolution of computer architectures, which would support timely updates of the platforms while providing the adequate motivations for all involved players.